Trabajo Fin de Grado
Ingeniería de Tecnologías de Telecomunicación

Análisis y capacidades de la tecnología Blockchain en el ámbito de la acreditación Académica.

Autor: David Félix Hernández Formento
Tutor: Federico José Barrero García

Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024

Trabajo Fin de Grado
Ingeniería de Tecnologías de Telecomunicación

Análisis y capacidades de la tecnología Blockchain en el ámbito de la acreditación Académica.

Autor:
David Félix Hernández Formento

Tutor:
Federico José Barrero García
Catedrático de Universidad

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2024

Trabajo Fin de Grado: Análisis y capacidades de la tecnología Blockchain en el ámbito de la acreditación Académica.

Autor: David Félix Hernández Formento

Tutor: Federico José Barrero García

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2024

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

En primer lugar, agradecer a mis padres, David y Soledad, ellos han sido los que me han apoyado desde el primer momento y durante todo momento, no solo durante la realización de este Trabajo de Fin de Grado, ni durante los años que he estado estudiando esta carrera, sino desde siempre. Desde que tengo memoria ellos han estado ahí para apoyarme y han hecho que no me falta de nada, me han escuchado y aconsejado, y he podido sentir todos los días el amor que tienen hacia mí, y espero que ellos también hayan podido sentir lo mucho que los aprecio y admiro. Muchas gracias nuevamente, os quiero.

También a mi abuela, José Fina, quien se ha preocupado por mis todas las semanas y todos los días sin falta, preguntándome como me iban los estudios y se alegraba por mis triunfos y me reconfortaba con mis fracasos. Muchas gracias abuela, te quiero.

A todos mis amigos y familiares que han hecho que estos últimos años sean más fáciles y llevaderos con todos los buenos momentos y aventuras que hemos tenido. Muchas gracias a todos.

Finalmente, agradecerle a mi tutor de este trabajo, Federico, no solo por su paciencia, sino también por las reuniones que hemos tenido con consejos no solo académicos y referentes a este trabajo, si no referentes a la vida. Muchas gracias.

David Félix Hernández Formento

Autor del Trabajo

Sevilla, 2024

Resumen

La tecnología Blockchain está cada vez más presente entre nosotros gracias a la popularización de las distintas monedas virtuales, especialmente con Bitcoin que desde su concepción en 2008 por Satoshi Nakamoto ha visto su auge. Este auge no sólo se ha visto en el mundo de estas denominadas "*criptos"*, si no también, numerosos investigadores han apostado por esta tecnología para solucionar problemas actuales como el de la privacidad, la falta de descentralización de la información y la seguridad. Se han iniciado numerosos proyectos para avanzar en distintos campos como gestión de la cadena de suministro, ciberseguridad, energía, industria automovilística e identidad digital entre otras.

Este trabajo tratará de analizar y obtener una vista general acerca de la herramienta Blockchain y sus utilidades, centrándose especialmente en el campo de la educación, además, se comentarán las características y la implementación de una red Blockchain como herramienta en el sector de la educación para acreditar de manera digital la superación de una asignatura de grado en la Universidad de Sevilla. Se acreditará la superación del alumno de la asignatura con una nota. Se explicará de manera detallada la estructura seguida, la tecnología escogida y las razones de su elección, la implementación y finalmente se discutirán los resultados obtenidos y de cómo se podría escalar y desarrollar la red en un futuro.

Abstract

Blockchain technology is increasingly present among us thanks to the popularization of different virtual currencies, especially with Bitcoin, which has seen its rise since its conception in 2008 by Satoshi Nakamoto. This boom has not only been seen in the world of these so-called “cryptos”, but also, numerous researchers have opted for this technology to solve current problems such as privacy, the lack of decentralization of information and security. Numerous projects have been started to advance in different fields such as supply chain management, cybersecurity, energy, the automotive industry and digital identity, among others.

This work will try to analyze and obtain a general view of the Blockchain tool and its utilities, focusing especially on the field of education. In addition, the characteristics and implementation of a Blockchain network as a tool in the education sector will be discussed. digitally accredit the completion of a degree subject at the University of Seville. The student's passing of the subject will be credited with a grade. The structure followed, the technology chosen and the reasons for its choice, the implementation will be explained in detail and finally the results obtained and how the network could be scaled and developed in the future will be discussed.

-translation by google-

Índice

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

"Strive not to be a success, but rather to be of value."

-Albert Einstein -

1.1 ¿Qué es la tecnología blockhain?

La tecnología Blockchain es una especia de libro mayor inmutable y compartido, que es usado para facilitar el proceso de constatar transacciones y rastrear bienes en una red de negocios. Esta tecnología se basa en una red peer-to-peer (p2p) y cada identidad recoge copias idénticas del libro mayor, siendo una copia que puede ser inspeccionada pero no controlada. Sólo se acepta un cambio si todas las entidades están de acuerdo, haciendo de esta manera innecesaria la actuación de un intermediario, es decir es una base de datos segura descentralizada.

La Blockchain utiliza como unidad más pequeña de información un denominado bloque, haciendo que la secuencia de estos bloques es a lo que nos referimos como Blockchain. Estos bloques contienen información de cada una de las transacciones y pueden tener información adicional como, quién, cómo, cuándo, cuánto o el estado de la transacción. Cada bloque está unido al anterior y al posterior, creando así la cadena de datos. En los bloques existe información del momento en que ocurrió la transacción para así poder ser unida de forma segura. Una vez un bloque haya sido validado por las distintas entidades y haya sido introducido en la cadena, es irreversible, haciendo que la introducción de cada bloque reafirme la verificación del bloque anterior y evitando así que existan copias modificadas de la cadena.

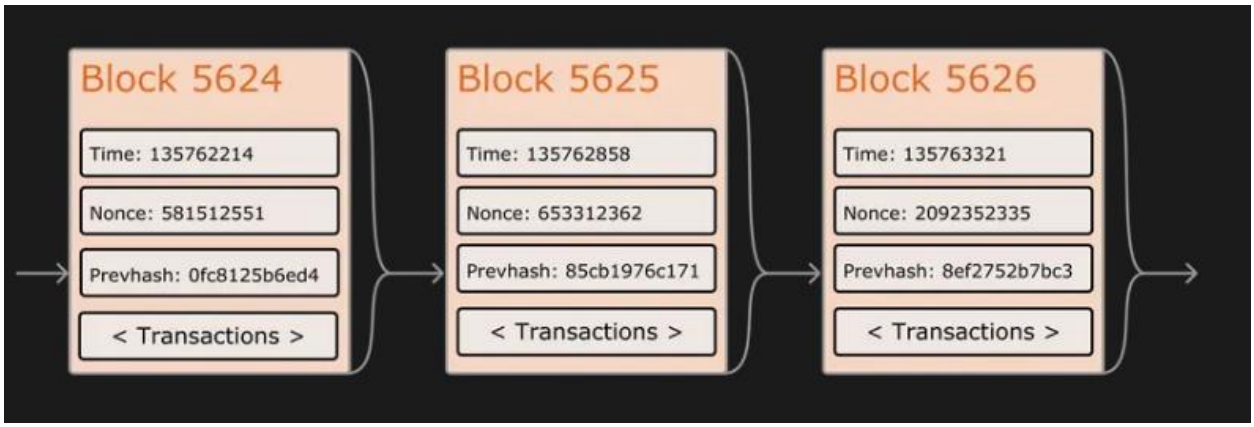


Figura 1. Estructura de la Blockchain. [1]

Una de las novedades que introduce Blockchain es el concepto de Dapp. Una Dapp, o Aplicación Descentralizada (Decentralized Application), es un tipo de aplicación que opera en una red descentralizada, como Blockchain, en lugar de depender de servidores centralizados. Estas aplicaciones están diseñadas para ser transparentes, seguras y resistentes a la censura, lo que significa que su funcionamiento no está controlado por una sola entidad. Las Dapps utilizan contratos inteligentes para ejecutar su lógica de negocio y pueden abarcar una variedad de funciones, desde sistemas financieros y de votación hasta juegos y redes sociales. Gracias a su naturaleza descentralizada, las Dapps ofrecen mayor transparencia, resistencia a la manipulación y autonomía, lo que las convierte en una opción atractiva para aquellos que buscan alternativas innovadoras a las aplicaciones centralizadas tradicionales.

1.2 Historia del blockchain

En 1991 emergió por primera vez el término Blockchain al ser acuñado por el científico y físico estadounidense W. Scott Stornetta y el criptógrafo Stuart Haber. Su trabajo inicial constaba de formar una red de bloques asegurada criptográficamente dónde nadie pudiese alterar las marcas de tiempo de los registros. En 1992, continuaron su trabajo consolidando los denominado Árbol de Merkle, técnica usada para aumentar la eficiencia de los bloques.

Sin embargo, no fue hasta 2008 cuando de verdad esta tecnología empezar a cobrar importancia cuando Satoshi Nakamoto usó la tecnología Blockchain como base para la creación de la primera criptomoneda, Bitcoin. Este usaba una red P2P para registrar las transacciones que usaban Bitcoin y añadiéndolas a la cadena de bloques. La primera transacción no se hizo hasta el 22 de Mayo de 2010 cuando se utilizó para pagar unas dos pizzas por el valor de 10 000 bitcoins.

[1] Imagen tomada de Ethereum Whitepaper | ethereum.org

Figura 2. Logo de Bitcoin. [2]

1.3 Ventajas e Inconvenientes de Blockchain

La tecnología Blockchain está en boca de todos en los últimos años y hay numerosas empresas que están apostando por ellas, llegando a tener una financiación de 2.6 miles de millones en el año 2022.[3]

A continuación, se van a listar una serie de ventajas que han hecho que Blockchain sea tan interesante a nivel mundial para los inversores y también cuales son las principales desventajas que señalan sus detractores.

Ventajas:

- Descentralización: como es de esperar la principal ventaja de la tecnología Blockchain es la eliminación de la figura de un intermediario, dando a los usuarios el control y poder total sobre sus datos. Esto es beneficioso en la manera en la que tradicionalmente los intermediarios se llevaban una comisión por cada transacción además de los intereses personales de los intermediarios que podían ofrecer servicios no tan honestos para su propio beneficio.
- Transparencia: Las transacciones son totalmente transparentes, cada nodo en la red tiene una copia de la cadena y todos los usuarios tienen acceso a estos datos. Los usuarios pueden darse cuenta de cambios en la cadena y ver como se actualiza, reforzando su seguridad.
- Seguridad: Al utilizar un sistema de consenso, todos los nodos deben de aceptar un cambio en la cadena para que este se efectúe. Además, el uso del Hash, una especie de huella digital única que identifica a cada bloque en la red, hace que los bloques estén encriptados correctamente y enlazados al nodo anterior, haciendo imposible que un usuario malicioso introduzca una transacción falsa o intente modificar una existente.
- Eficiencia: Blockchain elimina la necesidad de transacciones en papel y mediante intermediarios que requieren tiempo y en los que a veces ocurren errores humanos. Blockchain hace significativamente más rápido las transacciones y hace que los usuarios no deban tener registros de las diferentes transacciones ya que todos tienen acceso a ellas en un solo sitio común.
- Trazabilidad: Blockchain ofrece la capacidad de monitorizar las transacciones, haciendo fácil la trazabilidad por ejemplo de productos en una cadena de producción, además de ver como el producto ha ido pasando por las diferentes etapas de producción. Además, puede permitir a los compradores comprobar la autenticidad del producto y evitar el fraude.

[2] Imagen tomada de https://pixabay.com/es/vectors/search/bitcoin/
[3] Fuente https://www.statista.com/statistics/621207/worldwide-blockchain-startup-financing-history/

Inconvenientes:

- Rendimiento: Como uno de los puntos fuertes de la cadena es la redundancia en cada uno de los nodos, esto hace que cada nodo tenga que repetir el mismo proceso para actualizar su cadena y así tener una versión común para todos.
- Claves Privadas: Para hacer una transacción, el usuario necesita una clave privada. Esta clave es única y los demás usuarios no la pueden conocer. Esta clave debe mantenerse en un sitio seguro ya que, si se pierde acceso a ella, se perderá el acceso a tu usuario en la cadena junto con el acceso a tus transacciones y activos, ya que no hay manera de recuperarla.
- Uso de Energía: Debido a las operaciones de minería, que son las que se hacen en cada transacción a tiempo real y hacen que cada nodo tenga la misma información, Blockchain tiene un uso alto de energía y es uno de los principales puntos a desarrollar en esta tecnología.
- Almacenamiento: El problema del almacenamiento surge porque las bases de datos de Blockchain se mantienen permanentemente en todos los nodos de la red. No hay posibilidad de que los ordenadores personales puedan almacenar una cantidad infinita de datos que se siguen agregando a medida que aumenta el número de transacciones.

1.4 Usos de Blockchain

La naturaleza descentralizada de Blockchain hace que sea una de las mejores opciones para un amplio número de áreas distintas.

1. Sanidad: tras la pandemia hemos vivido bastantes situaciones difíciles en las que varios hospitales no tenían camas para todas las personas todo esto debido a un sistema centralizado que no promovía la comunicación entre distintos hospitales. Con un acercamiento descentralizado se podría conseguir una comunicación transparente, segura y de gran escalabilidad.

2. Gestión de cadena de suministro: la parte más importante de cualquier negocio es la de gestión de la cadena de suministro y ya sea un pequeño comercio minorista, un mayorista, o incluso un comercio electrónico tener una cadena de comercio optima puede ser la clave para un exitoso negocio. El punto débil actualmente de esta gestión se basa en la trazabilidad. Esto es fácilmente resuelto con el uso de Blockchain que proporciona una visión transparente de todo el proceso y la capacidad de rastrear los productos desde el origen hasta el punto final de entrega.

3. Usos gubernamentales: Hay varios usos para la Blockchain para usos de gobiernos como: la gestión de identidad, reemplazando el uso de documentos de identidad como los utilizados actualmente por identificación descentralizada con la capacidad de ser usada en cualquier lugar de manera segura sin los riesgos de suplantación; Para votar, de esta manera las votaciones pueden hacerse de manera transparente y segura evitando así problemas de falta de confianza del proceso por parte de los ciudadanos; impuestos, los gobiernos también pueden procesar impuestos a través de una plataforma descentralizada que brinda servicios gratuitos de pago de impuestos para todos

4. Web3: Internet es ya una parte inseparable de nuestras vidas y la integración de estos servicios en línea con Blockchain pueden suponer un cambio de paradigma para la llegada de Web3. Web3 es la tercera iteración de la red de redes que conocemos hoy en día, siendo la primera, Web1, la puesta en marcha de Internet a nivel mundial en la década de los 80 y la segunda, Web2, llegando con la capacidad de los usuarios para crear distintas cuentas en numerosas aplicaciones creando una identidad en línea lo que impulso las redes sociales y los comercios en internet. La Web3, es un cambio de paradigma que se centra en la inclusión de Internet de Las Cosas, IoT, Machine Learning e inteligencia artificial para crear nuevos servicios y

aplicaciones, a la vez que centrándose en la seguridad. La clave del Blockchain en esta nueva iteración es debido a que muchos de los nuevos protocolos que usarán estas nuevas aplicaciones y servicios son descentralizados y sería posible su interoperabilidad y automatización gracias a los contratos inteligentes brindados por Blockchain.

Además de estos existen otras numerosas áreas en las que existen usos para Blockchain como: la industria energética, finanzas y banca, propiedades e inmuebles, automoción, viajes, e innumerables más.

1.5 Blockchain en la educación.

Además de todos los usos comentados anteriormente, uno de los usos que está en auge y que tiene un gran potencial en los próximos años es en el sector de la educación. Como se ha comentado anteriormente, las características de Blockchain la hace idónea para sistema de acreditación y de identificación. De esta manera, se podría tener un registro de todos los estudiantes en una organización y tener acceso libre y descentralizado a las diferentes cualificaciones que estos han ido adquiriendo en su educación. Esto facilitaría procesos como el de un estudiante queriendo transferir su expediente de una universidad a otra.

Ya existen ejemplos de estos proyectos hoy en día, como el de OpenBlockchain[4] que junto a Open Univesity en Reino Unido, esperan desarrollar un nuevo tipo de universidad: una Universidad DAO. Una DAO es una Organización Autónoma Descentralizada que opera sin control central y sin humanos en el circuito y se basa en Contratos Inteligentes: piezas de código de ordenador en una cadena de bloques que pueden representar y promulgar contratos financieros y legales.

En una Universidad DAO, el valor y la reputación asociados con la enseñanza y el aprendizaje se contabilizarían a través de una cadena de bloques sin control central. En cambio, toda la comunidad de aprendizaje estaría de acuerdo en cómo se comparten y recompensan los elementos educativos, por ejemplo, los materiales de aprendizaje, los recursos didácticos y la enseñanza.

También existen proyectos para autenticación de certificados académicos, como el caso de Blockcerts, en el que Learning Machine (ahora parte de Hyland) se asoció con MIT Media Lab para crear Blockcerts, una plataforma estándar abierta para crear, emitir y verificar certificados respaldados por Blockchain. Al crear registros como expedientes académicos y credenciales en una cadena de bloques, la empresa puede revisar la credibilidad de los documentos y descubrir información falsificada. Los logros académicos (calificaciones, transcripciones e incluso diplomas) también se pueden almacenar en una cadena de bloques de Blockcerts para obtener una visión inmutable de la historia académica pasada.

[4] https://blockchain.open.ac.uk/

1.6 Objetivo

El objetivo del presente Trabajo de Fin de Grado será el de analizar y estudiar las capacidades actuales de Blockchain en un entorno educativo haciendo un repaso de la tecnología de y de sus utilidades generales en varios entornos, y sus aplicaciones actuales y casos de éxito, haciendo hincapié en el educativo. Y además, se va detallar las características de un sistema basado en Blockchain para acreditar la superación de una asignatura en la Universidad de Sevilla. Además, de un desarrollo inicial de dicho sistema y finalmente con pautas para poder escalar dicha cadena a más asignaturas y más grados para interconectar distintas entidades educativas con el fin de desarrollar un sistema de acreditación digital para estudios universitarios.

2 TECNOLOGÍA BLOCKCHAIN: APLICACIÓN EN ENTORNO EDUCATIVO

Blockchain es una tecnología revolucionaria de naturaleza descentralizada, segura y transparente que ha tenido el potencial de revolucionar las finanzas y que puede afectar de igual manera a la industria, la gestión de la cadena de suministro, la atención sanitaria o la educación. La integración de la tecnología blockchain en el sistema educativo tiene el potencial de mejorar en gran medida la eficiencia, seguridad y credibilidad del proceso educativo, creando plataformas seguras y transparentes para rastrear y verificar los logros académicos de los estudiantes [3]. La tecnología blockchain puede ayudar a crear un entorno más accesible y confiable del sistema educativo, que facilite que los estudiantes muestren sus habilidades y conocimientos a posibles empleadores.

Sin embargo, la tecnología blockchain ha tenido un desarrollo que podríamos denominar escaso en el ámbito educativo, bastante menor al alcanzado en otros ámbitos y, sinceramente, aún no podemos decir que se haya implantado de una manera eficaz. Uno de los principales beneficios que puede aportar esta nueva tecnología en el ámbito educativo es la creación y almacenamiento a prueba de manipulaciones y descentralizado de una transcripción digital del historial académico del estudiante, incluidas calificaciones, certificaciones y otros. Esto eliminaría la necesidad de transcripciones tradicionales en papel, que pueden perderse, dañarse o manipularse con relativa facilidad [4]. Otro beneficio potencial de blockchain en educación es la creación de una plataforma descentralizada para la emisión y verificación de documentos digitales y credenciales, lo que puede simplificar enormemente el proceso de obtención y verificación de las credenciales académicas al eliminar la necesidad de intermediarios. La propia naturaleza de la blockchain ayuda a prevenir el fraude, proporcionando una base confiable y un registro a prueba de manipulaciones de los logros de los estudiantes [5]. Así, existen interesantes aproximaciones relacionadas con el sistema Universitario [6], centradas en la mejora de los procedimientos para verificar la autenticidad de los títulos y diplomas. La educación y cursos on-line ofrecen un nicho de especial interés a la blockchain, al incorporar una plataforma segura y confiable para impartir y realizar un seguimiento y la certificación de los cursos, lo que puede ayudar a mejorar su credibilidad y reconocimiento por parte de los estudiantes y empleadores [7].

Blockchain representa una revolución tecnológica con potencial para modificar el proceso educativo, mejorando su eficiencia y credibilidad y creando plataformas seguras y transparentes para el seguimiento y verificación de los logros académicos de los estudiantes. Sin embargo, es necesario abordar numerosos desafíos para lograr el potencial pleno de esta tecnología en el sector educativo. Entre estos desafíos se encuentran la adopción propia de la tecnología, lo que requiere de un conocimiento técnico importante, la interoperabilidad y regulación del sistema que se desarrolle, su costo, escalabilidad y accesibilidad, o la imprescindible privacidad y seguridad de los datos almacenados. No se trata de retos sencillos de abordar. Así, el aspecto jurídico relacionado con una tecnología se caracteriza por la existencia de bloques que son registros de información y por la transparencia en la información que contienen. Los usuarios pueden acceder a la información de forma libre [8], lo que puede llegar a suponer un conflicto con el derecho a la protección de datos de carácter personal [9]. Otro conflicto jurídico que puede suscitarse aparece entre el principio de transparencia que caracteriza a la Blockchain y la confidencialidad o derecho a la privacidad del titular de los datos registrados mediante esta tecnología, para cuya convivencia pacífica la doctrina ha propuesto la seudonimización [10]. Del mismo modo, los datos almacenados son por lo general inmutables, lo que constituye un mecanismo de garantía de que nadie podrá alterarlos sin que el resto de los verificadores y titulares de los datos tengan conocimiento de ello. Sin embargo, la inmutabilidad de los datos, y su existencia perpetua en la red, entra en conflicto con el conocido "derecho al olvido" que ostentan los titulares sobre sus datos registrados en Internet e indexados en los motores de búsqueda **¡Error! No se encuentra el origen de la referencia.**

2.1 Casos de éxito en la educación

El uso de la tecnología blockckain en el ámbito educativo universitario no es nuevo, si bien aún se trata de una tecnología que no se encuentra ampliamente difundida. Existen diferentes universidades que han adoptado la tecnología Blockchain en sus programas educativos. Así, la Universidad de Nicosia (Chipre) ofrece un programa de maestría en Blockchain y Criptomonedas, permitiendo pagos en Bitcoin.

El MIT (Instituto de Tecnología de Massachusetts, en EEUU) emite certificados de finalización de cursos a través de una plataforma basada en Blockchain llamada Blockcerts. El College of London (UK) mejora la transparencia y la trazabilidad de la cadena de suministro de la universidad, garantizando el abastecimiento ético y sostenible de bienes y servicios, empleando VeChain. La Universidad de Melbourne (Australia) ha explorado el uso de la Blockchain para gestionar y proteger los datos de investigación, proporcionando un registro inmutable y transparente de las contribuciones de los investigadores. La Universidad de Stanford (EEUU) ha lanzado varios proyectos relacionados con la Blockchain, incluyendo el Centro de Investigación Blockchain de Stanford, que se centra en la investigación y el desarrollo de tecnologías Blockchain.

La propuesta pretende servir de ejemplo para la implementación y utilidad de la tecnología en la certificación educativa, con idea de extraer consecuencias sobre los beneficios acarreados para la institución educativa y para los estudiantes. Es esperable que la transparencia e integridad de los registros académicos mejorarán considerablemente, puesto que la naturaleza inmutable de la tecnología garantizará que las credenciales académicas emitidas (certificados y calificaciones) son auténticas y seguras, sin posibilidad de alteración o falsificación. En principio, esta característica debe proporcionar una mayor confianza a cualquier entidad interesada en la verificación de las credenciales de los estudiantes. Las transacciones se almacenan en una red descentralizada, lo que significa que no existe un punto aislado de los demás que los hackers puedan utilizar para modificar la red. Cada transacción está protegida con criptografía, lo que dificulta alterar o borrar registros. Esto hace que Blockchain sea perfecta para mantener seguros los registros de los estudiantes, que contienen información personal y confidencial relacionada con los logros académicos.

Por otro lado, la eficiencia en la gestión de registros académicos aumentaría notablemente. Con un sistema basado en Blockchain, el proceso de emisión, almacenamiento y verificación de certificados y notas se simplificaría y agilizaría significativamente. Los estudiantes podrían acceder a sus registros académicos de manera instantánea y segura, eliminando la necesidad de solicitar documentos físicos a las instituciones educativas. Esto no solo reduciría la burocracia y los costos administrativos asociados, sino que también proporcionaría un acceso más rápido y conveniente a la información académica. Además, la verificación se puede realizar en tiempo real con Blockchain, proporcionando eficiencia y rapidez al procedimiento.

Otro beneficio que puede considerarse clave de la tecnología Blockchain en la educación es la creación de identidades digitales para estudiantes, si bien debe solventar antes el derecho a la privacidad mencionado en la introducción de este trabajo. La identidad digital de un estudiante puede incluir información como su nombre, fecha de nacimiento y nivel académico, que se puede utilizar para seguir el progreso a lo largo de la carrera educativa e incluso profesional. Al disponer una identidad digital segura y descentralizada, los estudiantes pueden controlar quién tiene acceso a su información personal y es posible asegurar que los registros ni se pierden ni se alteran.

Finalmente, el uso de la tecnología blockchain en la educación también puede ayudar con la distribución de recursos educativos, como pueden ser las becas y subvenciones. Al tener un libro de transacciones seguro, las instituciones educativas pueden realizar un seguimiento de la distribución de los recursos, garantizando que se utilicen para el fin previsto. El uso de contratos inteligentes puede automatizar la distribución de recursos, reduciendo el riesgo de error humano y aumentando la eficiencia del proceso. Nuevamente, será necesario

solventar antes el derecho a la privacidad y al olvido digital de las personas que intervengan en el sistema que se desarrolle

2.2 Desafíos de Blockchain en la educación

Aunque la integración de la tecnología Blockchain en la educación tiene un gran potencial, es importante reconocer las limitaciones y desafíos que enfrenta esta tecnología. Aquí están algunas de las principales limitaciones del Blockchain en la educación:

1. Adopción: Uno de los mayores desafíos es la necesidad de una adopción generalizada. Esto implica la participación de estudiantes, educadores, instituciones y empleadores, todos deben estar dispuestos a aceptar la tecnología para aprovechar sus beneficios.
2. Conocimiento técnico: Otro desafío es la necesidad de que las personas y organizaciones comprendan bien la tecnología para implementarla y usarla eficazmente. Esto incluye entender los detalles técnicos del Blockchain, así como las implicaciones de seguridad y privacidad.
3. Interoperabilidad: La tecnología Blockchain aún está en sus primeras etapas de desarrollo y no hay estandarización o interoperabilidad entre diferentes plataformas Blockchain. Esto dificulta que las instituciones la usen de manera uniforme.
4. Regulación: Actualmente hay pocas regulaciones sobre el uso del Blockchain en la educación, lo que puede dificultar su implementación efectiva, especialmente en temas como la privacidad y seguridad de los datos.
5. Costo: Implementar y mantener la tecnología Blockchain puede ser costoso. Esto incluye los costos de hardware, software, personal y capacitación.
6. Privacidad y seguridad de los datos: Asegurar la privacidad y seguridad de los datos educativos sensibles es un gran desafío. Esto requiere desarrollar sistemas seguros y procesos para proteger los datos de los estudiantes y asegurar que las personas tengan control sobre sus propios datos.
7. Escalabilidad: La tecnología Blockchain debe ser capaz de manejar grandes cantidades de datos para ser efectiva en el sector educativo. Esto requiere el desarrollo de sistemas escalables.
8. Integración con sistemas existentes: Integrar la tecnología Blockchain con los sistemas educativos actuales es un gran desafío. Esto implica desarrollar sistemas y procesos que puedan integrar eficazmente el Blockchain sin comprometer la integridad y seguridad de los datos educativos.
9. Estandarización: La necesidad de estandarización y consistencia en el uso de la tecnología Blockchain en la educación es esencial. Esto requiere el desarrollo de estándares y protocolos comunes.
10. Educación y capacitación: Educar y capacitar a las personas en el uso de la tecnología Blockchain es esencial para su implementación exitosa. Esto incluye desarrollar programas de capacitación y materiales educativos.
11. Confianza y fiabilidad: La confianza y fiabilidad son factores críticos. Esto requiere desarrollar sistemas seguros y fiables que puedan ser confiados por individuos y organizaciones en el sector educativo.
12. Verificación y validación: Verificar y validar la autenticidad y precisión de los datos educativos es un gran desafío. Esto requiere sistemas seguros para verificar y validar los datos, y métodos para asegurar que los datos permanezcan precisos e inalterables.
13. Gestión de datos: La gestión efectiva de los datos es esencial. Esto implica desarrollar sistemas y procesos para gestionar y organizar los datos educativos, asegurando su seguridad y accesibilidad.
14. Accesibilidad: Asegurar que los estudiantes y educadores tengan acceso a la tecnología educativa.

La implementación de la tecnología Blockchain en la educación requiere un entorno completo y de apoyo que pueda manejar los datos y transacciones complejas asociadas con esta tecnología emergente. Este entorno

incluye una variedad de hardware, software, dispositivos, servicios y soluciones de seguridad que trabajen juntos para asegurar el despliegue exitoso del Blockchain en la educación.

Por lo tanto, aunque es importante reconocer los desafíos asociados con la adopción del Blockchain en la educación, también es esencial reconocer los beneficios potenciales y las características únicas que esta tecnología puede ofrecer al sector educativo. Al abordar estos desafíos y aprovechar los beneficios del Blockchain, las instituciones educativas pueden mejorar la sostenibilidad educativa y asegurar sus datos y certificaciones.

2.3 Mecanismos de Consenso en Blockchain y su Evolución

1. Prueba de Trabajo (PoW - Proof of Work)

PoW es el primer y más conocido mecanismo de consenso utilizado por Bitcoin. Los participantes (mineros) compiten para resolver complejos problemas matemáticos, y el primero en resolverlo añade un nuevo bloque a la cadena y recibe una recompensa.

Ofrece, Alta seguridad y resistencia a ataques y descentralización fuerte. Sin embargo, tiene un alto consumo de energía. Además, es ento y costoso en términos de recursos.

2. Prueba de Participación (PoS - Proof of Stake)

En PoS, los validadores son elegidos para crear nuevos bloques y validar transacciones según la cantidad de criptomoneda que poseen y están dispuestos a "apostar" o poner en juego.

Tiene un menor consumo de energía que PoW, y mayor velocidad y eficiencia en transacciones.

Por otro lado, tiene un riesgo de centralización (los más ricos tienen más poder). Y finalemte ,este mecanismo es menos probado en comparación con PoW.

3. Delegated Proof of Stake (DPoS)

DPoS es una variación de PoS donde los tenedores de monedas eligen delegados que validan transacciones y mantienen la red. Ejemplo: EOS.

Ofrece, Alta eficiencia y velocidad, mayor escalabilidad. Sin embargo, como desventajas, presenta un mayor riesgo de centralización y dependencia de la elección de delegados.

4. Prueba de Autoridad (PoA - Proof of Authority)

PoA selecciona validadores basándose en su reputación y autoridad. Comúnmente usado en redes privadas y permisos. Ejemplo: VeChain.

Ofrece como ventajas, alta eficiencia y bajo consumo de energía; adecuado para aplicaciones empresariales y privadas. Pero como desventajas, centralización significativa, dependencia en la reputación de los validadores.

5. Prueba de Capacidad (Proof of Capacity) y Prueba de Espacio-Tiempo (Proof of Space-Time)

En estos mecanismos, los participantes utilizan su capacidad de almacenamiento para minar bloques. Ejemplo: Chia (Proof of Space-Time).

Sus ventajas son: menor consumo energético comparado con PoW, utilización de recursos de almacenamiento.

Y sus desventajas: requiere gran capacidad de almacenamiento, no tan probada y utilizada como PoW o PoS.

6. Mecanismos de Consenso Híbridos

Los mecanismos de consenso híbridos combinan elementos de diferentes métodos de consenso para aprovechar sus fortalezas y mitigar sus debilidades. Estos sistemas buscan un equilibrio entre seguridad, eficiencia y descentralización. Por ejemplo, una red podría usar Proof of Work (PoW) para la creación de bloques iniciales y Proof of Stake (PoS) para la validación de transacciones, logrando así un balance entre la alta seguridad de PoW y la eficiencia energética de PoS. Estos enfoques permiten una mayor flexibilidad y adaptación a diferentes necesidades y contextos.

7. Consenso BFT en Consorcio

El consenso Byzantine Fault Tolerance (BFT) es un mecanismo que permite a un sistema continuar operando correctamente incluso si algunos de sus nodos fallan o actúan de manera maliciosa. En un entorno de consorcio, donde un grupo de organizaciones gestiona conjuntamente una red blockchain, el consenso BFT es particularmente útil. Los nodos del consorcio, que ya se conocen y confían entre sí hasta cierto punto, usan BFT para validar transacciones y crear nuevos bloques. Esto mejora la eficiencia y velocidad del consenso, ya que no se requiere la misma cantidad de prueba de trabajo o participación que en redes totalmente públicas y anónimas.

8. Consenso BFT Híbrido

El consenso BFT híbrido combina el enfoque de tolerancia a fallos bizantinos con otros mecanismos de consenso para mejorar la robustez y eficiencia de la red. Un ejemplo típico podría ser la combinación de BFT con Proof of Stake (PoS), donde los validadores se seleccionan y rotan usando PoS, pero la validación de bloques sigue un protocolo BFT. Este enfoque híbrido busca proporcionar una mayor resistencia a fallos y ataques, optimizando al mismo tiempo la eficiencia energética y la velocidad de las transacciones. Esto es especialmente valioso en redes que necesitan un alto grado de seguridad y rendimiento, como las financieras o de grandes consorcios empresariales.

En este trabajo se hará uso de tokens de tipo NFT usando la red Ethereum, en la red Ethereum, los NFTs (tokens no fungibles) utilizan el mecanismo de consenso Proof of Stake (PoS) desde la actualización conocida como "The Merge" que se completó en septiembre de 2022. Antes de esta actualización, Ethereum utilizaba Proof of Work (PoW).

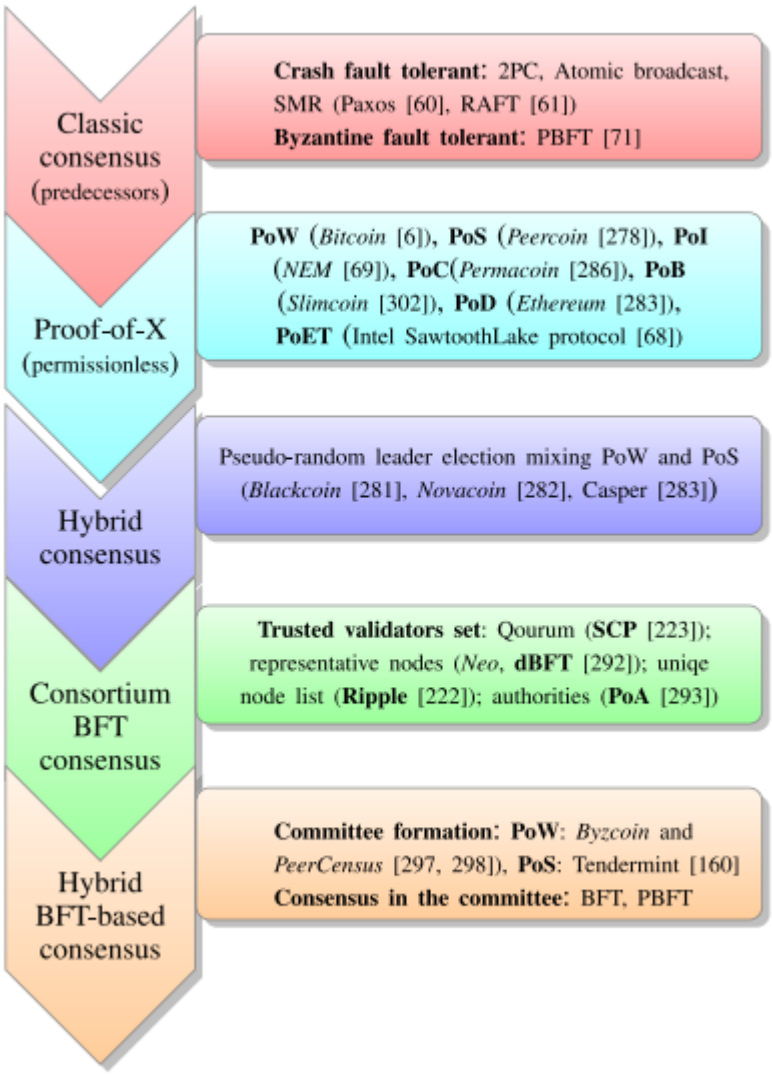


Figura 3. Resumen y evolución de los mecanismos de consenso. [5]

2.4 Entorno y Equipamiento para Blockchain en la Educación

La implementación de la tecnología Blockchain en la educación requiere un entorno complejo que incluye una variedad de hardware, software, dispositivos, servicios y soluciones de seguridad. Las organizaciones deben considerar cuidadosamente estos requisitos al planificar la implementación del Blockchain para garantizar el éxito de sus iniciativas. Esta tecnología innovadora necesita una amplia gama de equipos para asegurar la transferencia eficiente y segura de datos y transacciones educativas dentro de la red Blockchain.

[5] Imagen tomada de [1] M. Belotti, N. Božić, G. Pujolle, S. Secci, "A Vademecum on Blockchain Technologies: When, Which, and How," IEEE Communications Surveys & Tutorials, vol. 21, no. 4, pp. 3796-3838, 2019.

- Hardware: Los requisitos de hardware para Blockchain en la educación son exigentes, ya que la tecnología necesita recursos informáticos potentes y confiables para manejar grandes cantidades de datos y transacciones. Esto incluye servidores, dispositivos de almacenamiento y otros componentes de hardware que puedan almacenar, procesar y transmitir datos de manera eficiente. El hardware debe ser escalable, seguro y capaz de manejar cantidades crecientes de datos a medida que crece la red Blockchain.
- Software: El entorno de software es tan importante como el hardware en la implementación del Blockchain en la educación. La tecnología Blockchain requiere una gama de plataformas y herramientas de software para gestionar los datos y transacciones en la red. Esto incluye plataformas Blockchain como Ethereum, Hyperledger y otras soluciones similares, así como herramientas para gestionar los datos y transacciones en la red Blockchain.
- Dispositivos: Para que las personas y organizaciones participen en una red Blockchain, deben tener acceso a dispositivos como computadoras, smartphones y otros dispositivos conectados a internet. Estos dispositivos deben ser seguros, confiables y estar equipados con el software y las aplicaciones necesarias para acceder a la red Blockchain y a los datos educativos.
- Conectividad a Internet: La conectividad a internet rápida y confiable es esencial para el uso efectivo de la tecnología Blockchain en la educación. Esto permite a las personas y organizaciones participar en la red Blockchain y acceder a los datos y recursos educativos desde cualquier lugar y en cualquier momento.
- Servicios en la Nube: Muchas instituciones y organizaciones están recurriendo a servicios basados en la nube para la implementación de la tecnología Blockchain en la educación. Esto les permite aprovechar los beneficios de escalabilidad y seguridad de la nube, al mismo tiempo que reducen los costos asociados con la infraestructura de hardware y software. Los servicios en la nube también pueden proporcionar un entorno más flexible y adaptable para implementar la tecnología Blockchain.
- Proveedores de Blockchain como Servicio (BaaS): Los proveedores de BaaS ofrecen soluciones basadas en la nube para la implementación de la tecnología Blockchain. Esto incluye servicios como el hospedaje y la gestión de redes Blockchain, así como el desarrollo de soluciones Blockchain personalizadas para necesidades educativas específicas. Los proveedores de BaaS pueden ofrecer a las organizaciones una forma más eficiente y rentable de implementar la tecnología Blockchain en la educación.
- Soluciones de Seguridad: Asegurar la privacidad y seguridad de los datos educativos es esencial al implementar la tecnología Blockchain en la educación. Esto requiere el uso de soluciones de seguridad confiables y seguras, como la encriptación, firewalls y sistemas de detección de intrusos. El entorno de seguridad también debe incluir planes de recuperación ante desastres y de continuidad del negocio para asegurar que los datos educativos permanezcan disponibles y seguros incluso en caso de crisis.

La implementación de la tecnología Blockchain en la educación necesita un entorno complejo y completo, incluyendo hardware, software, dispositivos y soluciones de seguridad. Es crucial que las organizaciones consideren cuidadosamente estos requisitos para asegurar el éxito de sus iniciativas y la transferencia eficiente y segura de datos y transacciones educativas dentro de la red Blockchain.

2.5 Ethereum

2.5.1.1 ¿Qué es Ethereum?

Ethereum es una plataforma descentralizada que permite a los desarrolladores construir y desplegar contratos inteligentes y aplicaciones descentralizadas (dApps). Fue propuesta por Vitalik Buterin en 2013 y lanzada en 2015. Su criptomoneda nativa, Ether (ETH), se utiliza para pagar transacciones y servicios dentro de la red. Ethereum se diferencia de Bitcoin al ofrecer una plataforma más versátil, permitiendo a los desarrolladores crear aplicaciones complejas sobre su Blockchain.

2.5.1.2 ¿Cómo funciona Ethereum?

Ethereum utiliza una Blockchain pública y descentralizada similar a la de Bitcoin, pero con capacidades adicionales para ejecutar contratos inteligentes.

- Contratos Inteligentes: Son programas autoejecutables donde los términos del acuerdo están escritos en código. Estos contratos se ejecutan automáticamente cuando se cumplen ciertas condiciones, eliminando la necesidad de intermediarios y reduciendo el riesgo de fraude.

- Gas: Para ejecutar operaciones en la red Ethereum, se requiere una cantidad de "gas". El gas es una medida de la cantidad de trabajo computacional necesario para realizar transacciones y ejecutar contratos inteligentes. Los usuarios pagan con Ether para cubrir el costo del gas.

- Minería y Consenso: Inicialmente, Ethereum utilizaba Proof of Work (PoW) para validar transacciones y asegurar la red. Sin embargo, Ethereum está en proceso de transición a Proof of Stake (PoS) a través de una serie de actualizaciones llamadas Ethereum 2.0, que buscan mejorar la escalabilidad y la eficiencia energética de la red.

2.5.1.3 Ethereum Virtual Machine (EVM)

La Ethereum Virtual Machine (EVM) es el motor que ejecuta contratos inteligentes en la Blockchain de Ethereum. La EVM es una máquina virtual completa de Turing, lo que significa que puede ejecutar cualquier script complejo que un ordenador pueda procesar. Todos los nodos de la red ejecutan la EVM, asegurando que las transacciones y contratos inteligentes se ejecuten de manera consistente en toda la red.

2.5.1.4 Aplicaciones Desarrolladas con Ethereum

Ethereum ha dado lugar a una amplia gama de aplicaciones descentralizadas en varios campos:

- Finanzas Descentralizadas (DeFi): Incluye aplicaciones que permiten realizar préstamos, intercambios de activos, y obtener intereses de manera descentralizada. Ejemplos incluyen Uniswap (un intercambio de criptomonedas descentralizado), Compound (plataforma de préstamos y ahorros), y Aave.

- Tokens No Fungibles (NFTs): Los NFTs son tokens únicos que representan la propiedad de un activo digital. Plataformas como OpenSea y Rarible permiten la creación, compra y venta de NFTs, usados en arte digital, coleccionables, y más.

- Organizaciones Autónomas Descentralizadas (DAOs): Las DAOs son organizaciones gestionadas por

contratos inteligentes y gobernadas por los miembros a través de tokens de gobernanza. Un ejemplo notable es MakerDAO, que gobierna el stablecoin DAI.

- Juegos y Coleccionables: Juegos como CryptoKitties permiten a los jugadores criar, comprar y vender gatos digitales únicos, demostrando el potencial de los NFTs en los juegos.

Figura 4. Logo de MakerDao.[6]

[6] Imagen tomada de https://cryptoplaza.es/viejo/integra-dai-y-trabaja-con-makerdao-en-tu-proyecto/

3 CARACTERÍSTICAS DEL SISTEMA DE CERTIFICACIÓN EDUCATIVO DESARROLLADO

El sistema de certificación académica planteado utiliza la tecnología Blockchain con NFT como solución innovadora diseñada para mejorar la gestión y verificación de credenciales académicas en la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla, aprovechando las fortalezas de la tecnología Blockchain para garantizar la seguridad, transparencia e integridad de los registros académicos, y permitiendo a los estudiantes demostrar sus logros de manera confiable.

El sistema evalúa, mediante un contrato inteligente y de manera automatizada, los diferentes factores que intervienen en la calificación del alumno, limitando la intervención del profesorado a las acciones estrictamente necesarias (evaluaciones de tipo cualitativo que requieran de cuantificación). Este enfoque no solo simplifica la gestión de evaluaciones, sino que empodera a los alumnos al proporcionarles un entorno predefinido e inalterable de evaluación, así como certificados digitales sólidos y verificables de las asignaturas que han cursado. Se garantiza la transparencia, rapidez e inmutabilidad de las calificaciones y se consigue eliminar barreras burocráticas para la obtención de certificaciones académicas, fortaleciéndose la credibilidad del curriculum vitae de los estudiantes.

Se emplea la red Ethereum, una plataforma de blockchain pública y descentralizada, ampliamente conocida por su soporte de contratos inteligentes. Se han elegido herramientas como Remix e IDE para el desarrollo de los contratos inteligentes, así como el lenguaje de programación Solidity para la implementación de éstos. Además, se utiliza un servicio de almacenamiento IPFS para asegurar la integridad y disponibilidad de los datos fuera de la cadena.

A continuación, se enumeran los pasos seguidos en el proceso de acreditación realizado:

1) Creación del contrato inteligente. Un profesor crea un contrato inteligente en la blockchain que define las reglas y condiciones para la emisión de calificaciones a los estudiantes. Este contrato inteligente actúa como un intermediario confiable que garantiza la ejecución de las condiciones acordadas de manera automatizada y segura.
2) Generación del token para el estudiante. Cuando el profesor califica a un estudiante en una asignatura, genera un archivo JSON que contiene información relevante como la nota y el nombre del alumno, la asignatura o cualquier otro detalle que se considere pertinente. Este archivo JSON actúa como el certificado digital de la calificación que se propone.
3) Subida del archivo JSON a IPFS (sistema de archivos interplanetario). El archivo JSON generado se sube a IPFS, un sistema de archivos descentralizado que utiliza la tecnología de la cadena de bloques para almacenar y distribuir contenido de manera descentralizada. IPFS asigna un identificador único al archivo, conocido como CID (Content IDentifier), que puede utilizarse para acceder al archivo desde cualquier lugar de la red IPFS.
4) Almacenamiento del CID en Pinata. Pinata es un servicio que permite almacenar y gestionar archivos en IPFS de manera conveniente. El profesor utiliza Pinata para subir el archivo JSON a IPFS y obtiene un token único de Pinata que representa el archivo subido.
5) Entrega del CID al estudiante. Una vez que el archivo JSON está almacenado en IPFS y se ha obtenido el token de Pinata, el profesor entrega este token al estudiante. El token actúa como prueba criptográfica de la existencia e integridad del certificado de calificación del estudiante en IPFS.
6) Reclamación del certificado por parte del estudiante. El estudiante puede reclamar su certificado de

calificación utilizando el token recibido del profesor. Al presentar este token, el estudiante puede acceder al archivo JSON almacenado en IPFS y verificar la autenticidad de su calificación.

3.3 Elecciones Tecnológicas

El auge de la tecnología Blockchain en los últimos años ha generado un gran interés en la creación de redes descentralizadas y seguras. Esta tecnología ha dado lugar a una variedad de enfoques y plataformas para construir y gestionar redes Blockchain, cada una con sus propias características, ventajas y desafíos. Desde la popularidad de Ethereum y Hyperledger Fabric hasta la emergencia de nuevas cadenas de bloques como Polkadot y Solana, el panorama de las tecnologías Blockchain ha experimentado una rápida evolución. Este constante progreso se debe en gran medida a la demanda de soluciones innovadoras para problemas de confianza, seguridad y eficiencia en diversas industrias. La investigación y el desarrollo continuos en el campo de la tecnología Blockchain están impulsando nuevas mejoras y funcionalidades, lo que garantiza que esta tecnología siga siendo una fuerza disruptiva y en constante evolución en los próximos años.

Para la elección de la red Blockchain a utilizar en la propuesta presentada en este trabajo, se estudiaron como posibilidades Ethereum (ETH), Hyperledger Fabric (FABRIC), Solana (SOL) y Polkadot (DOT), ofreciendo cada red una serie de ventajas e inconvenientes que se ilustran en la Tabla 1. Se ha optado por la red Ethereum por las diferentes herramientas que ofrece, la versatilidad en ejecución de los contratos inteligentes, así como la importante comunidad y madurez en el sector que aporta (ampliamente adoptada y sólido ecosistema de desarrollo). Además, Ethereum ofrece la capacidad de crear contratos inteligentes utilizando su propia criptomoneda, lo que permite proporcionar un incentivo económico a los participantes en el sistema.

Tabla 1. Ventajas e inconvenientes en la implantación de un sistema de certificación curricular.

Red	**Ventajas**	**Inconvenientes**
ETH	Versatilidad en la ejecución de los contratos inteligentes. Gran comunidad y ecosistema de desarrollo. Seguridad robada con Prueba de Trabajo (PoW).	Escalabilidad limitada con congestión de red. Privacidad estándar, menos adecuada para casos sensibles.
FABRIC	Enfoque empresarial con características avanzadas. Privacidad granular y control de acceso. Escalabilidad y eficiencia para entornos empresariales.	Mayor complejidad de configuración. Comunidad más pequeña en comparación con Ethereum.
SOL	Alta velocidad y escalabilidad hasta 65,000 TPS. Tarifas de transacción bajas y rendimiento elevado. Completo soporte Turing, adecuado para aplicaciones complejas.	Menor madurez y riesgo potencial. Comunidad y ecosistema en crecimiento, pero aún menor.
DOT	Interoperabilidad entre múltiples cadenas de bloques. Escalabilidad mediante conexión de cadenas paralelas.	Menor adopción y madurez en comparación con Ethereum. Mayor complejidad técnica para desarrolladores.

Para seleccionar el entorno de desarrollo también se estudiaron tres opciones de IDE (*Integrated Development Environment*): Remix, Truffle y VS Code con extensiones para Ethereum. El resumen de las ventajas e inconvenientes de los IDE analizados se muestran en la Tabla 2.

Tabla 2. Ventajas e inconvenientes de diferentes IDE analizados.

IDE	Ventajas	Inconvenientes
Remix	Fácil de usar y accesible en línea. Soporte para el desarrollo y despliegue de contratos inteligentes en Ethereum. Amplia comunidad y documentación.	Limitaciones en funcionalidades avanzadas en comparación con IDE locales. Dependencia de conexión a internet.
Truffle	Potente suite de desarrollo para Ethereum. Soporte para desarrollo local y pruebas Herramientas avanzadas para testing, migraciones y despliegue de contratos.	Curva de aprendizaje más pronunciada para principiantes. Requiere configuración local y dependencias adicionales. Menos accesible para desarrolladores novatos.
VS Code	Integración con el popular editor de código Visual Studio Code. Personalización y soporte de la comunidad.	Requiere configuración adicional y descarga de extensiones. Menor soporte específico para Ethereum en comparación con Remix.

Finalmente se escogió Remix debido a su facilidad de uso y su integración directa con la red Ethereum, lo que facilita el desarrollo y la depuración de contratos inteligentes. Además, está basado en la web y permite escribir, probar y desplegar contratos inteligentes en la red Ethereum de manera muy sencilla.

En el caso del lenguaje de programación empleado para el desarrollo de los contratos inteligentes, se han contemplado las alternativas de Solidity y Vyper, cuyas ventajas e inconvenientes se muestran en la Tabla 3.

Tabla 3. Ventajas e inconvenientes de los diferentes lenguajes de programación para el desarrollo de los contratos inteligentes analizados.

Lenguaje	Ventajas	Inconvenientes
Solidity	Diseñado específicamente para contratos inteligentes en Ethereum. Amplia adopción y comunidad activa. Integración con herramientas de desarrollo como Remix y Truffle.	Posibles vulnerabilidades y errores de seguridad debido a su complejidad. Limitaciones en comparación con lenguajes más maduros.
Vyper	Lenguaje más seguro y fácil de entender que Solidity. Enfoque en la simplicidad y la seguridad. Menos propenso a errores y vulnerabilidades de seguridad.	Menor adopción y comunidad en comparación con Solidity. Limitaciones en funcionalidades avanzadas. Menos herramientas de desarrollo y soporte disponible.

La opción de Vyper se descartó debido a su menor adopción y por sus limitaciones frente a Solidity. Además, Solidity se utiliza de manera bastante habitual en el ecosistema Ethereum y se integra de manera directa con Remix, estando diseñado específicamente para la creación de contratos inteligentes, lo que lo hace adecuado para este caso de uso.

Finalmente, para la elección del servicio de almacenamiento IPFS (InterPlanetary File System) se barajaron como opciones Pinata, Infura o Textile. En este caso, las ventajas e inconvenientes que planteaban para el desarrollo del sistema a desarrollar se han descrito en la Tabla 4, optando por el uso de Pinata gracias a su fácil configuración y creación de usuario, además de su intuitiva interfaz de usuario. Por otro lado, Pinata ofrece un plan gratuito que cumple todas las necesidades del proyecto dado que el sistema planteado es un ejemplo de implementación a nivel de asignatura que no va a necesitar de grandes volúmenes de almacenamiento de datos.

Tabla 4. Ventajas e inconvenientes de los diferentes servicios de almacenamiento IPFS analizados.

Servicio IPFS	**Ventajas**	**Inconvenientes**
Pinata	Fácil de usar y configurar para almacenamiento IPFS. Interfaz de usuario intuitiva y documentación detallada. Características adicionales como análisis de archivos y gestión de tokens.	Planes de precios costosos para grandes volúmenes de almacenamiento.
Infura	Amplia red de nodos IPFS para una mayor disponibilidad y redundancia.	Limitaciones en el plan gratuito, como cuotas de uso y acceso restringido a algunas características.
Textile	Herramientas y bibliotecas adicionales para desarrolladores, incluyendo Textile Buckets y Threads.	Curva de aprendizaje para principiantes en el uso de herramientas y bibliotecas Textile. Menor alcance y adopción en comparación con alternativas más establecidas.

4 DESARROLLO E IMPLEMENTACIÓN DEL SISTEMA DE CERTIFICACIÓN EDUCATIVA

A la hora de desarrollar el sistema de certificación educativa e implementarlo se hizo hincapié en 3 puntos:

1. La creación del Smart Contract usando Solidity mediante Remix
2. La creación de una cuenta de usuario en un sistema de IPFS como Pinata y la subida de los archivos necesarios para poder gestionar los certificados académicos de los alumnos.
3. La capacidad del alumno para recibir sus cetificados mediante una Dapp usando Remix y la capacidad de visualizar sus certificados mediante la plataforma OpenSea.

4.1 Introducción al lenguaje Solidty

Solidity es un lenguaje de programación reciente desarrollado por Ethereum, una de las criptomonedas más importantes por capitalización de mercado. Se lanzó en 2015 bajo el liderazgo de Christian Reitwiessner. Sus características clave incluyen su naturaleza de alto nivel diseñada específicamente para la implementación de contratos inteligentes, así como su orientación a objetos, o más precisamente, a contratos.

Este lenguaje ha sido fuertemente influenciado por otros como Python, C++ y JavaScript. Se ejecuta en la Ethereum Virtual Machine (EVM), que es un componente central de la infraestructura de Ethereum. Solidity es altamente versátil y admite la programación definida por el usuario, bibliotecas y herencia complejas. Es el lenguaje principal utilizado para plataformas que se ejecutan en Blockchain.

Ethereum, por su parte, es una plataforma descentralizada de código abierto basada en Blockchain. Su función principal es ejecutar contratos inteligentes, es decir, aplicaciones que ejecutan programas exactamente como fueron programados, sin posibilidad de fraude, interferencia de terceros, censura o tiempo de inactividad. La Ethereum Virtual Machine (EVM) es fundamental en este proceso, ya que proporciona un entorno de ejecución seguro para estos contratos.

Los contratos inteligentes, escritos en Solidity, son programas de alto nivel que se compilan en bytecode de EVM y se despliegan en la Blockchain de Ethereum para su ejecución. Esto permite la ejecución de transacciones confiables y rastreables sin la necesidad de una autoridad central.

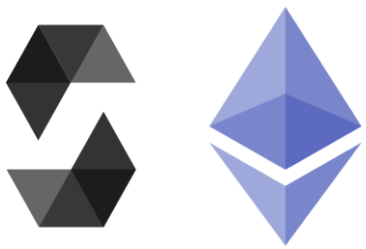


Figura 5. Logo de Solidity, izquierda, y de Ethereum, derecha[7]

[7] Imagen tomada de: https://blog.stackademic.com/understanding-solidity-a-beginners-guide-to-ethereum-smart-contracts-82c234f57b6b

Existen dos formas de ejecutar un contrato inteligente en Solidity:

1) Modo Offline: Ejecutar un contrato inteligente en Solidity en modo offline requiere tres requisitos previos y 4 pasos principales que deben seguirse para que el contrato inteligente se ejecute:

Requisitos previos: Descargar e instalar node.js. Instalar Truffle globalmente. Instalar ganache-cli.

Objetivos:

- Crear un proyecto de Truffle y configurar una red de desarrollo.
- Crear y desplegar contratos inteligentes.
- Interactuar con el contrato inteligente desde la consola de Truffle.
- Escribir pruebas para probar las principales características ofrecidas por Solidity.

2) Modo Online: El IDE de Remix se utiliza generalmente para compilar y ejecutar contratos inteligentes en Solidity en modo online.

Para este trabajo se ha utilizado el modo Online ya que es más accesible y fácil de usar, a la vez que trae más ventajas para poder desplegar el contrato desde cualquier lugar y para facilitar que nuestro contrato se comporte como una Dapp.

4.2 Creación del Smart Contract

Para crear el token Educativo, en nuestro caso llamado EDUToken, se ha basado en los tokens NFT que ya existen en la actualidad en las redes Ethereum debido a sus ventajas y características que las hacen muy afines a las necesidades de nuestros certificados académicos.

Un NFT, o token no fungible (del inglés Non-Fungible Token), es un tipo de activo digital único e indivisible que se registra en una Blockchain. A diferencia de los tokens fungibles como criptomonedas o tokens de utilidad, que son intercambiables entre sí en términos de valor, los NFTs son únicos y no pueden ser intercambiados uno a uno sin pérdida de valor o significado.

La norma ERC-721 es uno de los estándares más comunes para la creación de NFTs en la red Ethereum. Esta norma define un conjunto de reglas y funciones para la creación, transferencia y gestión de tokens no fungibles. Con la norma ERC-721, cada token tiene un identificador único y es indivisible, lo que significa que no se pueden dividir en partes más pequeñas.

Esta norma ERC-721 está descrita en una biblioteca de contratos inteligentes llamada OpenZepelin como un estándar para la red Ethereum. OpenZeppelin proporciona una amplia gama de contratos inteligentes pre-auditados y probados que cubren diversas funcionalidades comunes, como la gestión de tokens, la emisión de NFTs, la gobernanza y más.

Los NFT utilizan lo que es denominado proof of ownership. El proof of ownership (prueba de propiedad) en el contexto de los NFTs se refiere a la capacidad de un usuario para demostrar que es el propietario legítimo de un activo digital único en la Blockchain. Esto se logra mediante la firma criptográfica del propietario sobre una transacción específica que transfiere el NFT de una dirección a otra. El contrato inteligente que gestiona el NFT verifica esta firma y registra el cambio de propiedad en la Blockchain.

La principal diferencia entre el proof of ownership utilizado en los NFTs y otros mecanismos de prueba, como el proof of work (prueba de trabajo) o el proof of stake (prueba de participación), es que el proof of ownership se centra en demostrar la posesión de un activo digital único en lugar de demostrar la contribución de trabajo computacional o la participación en la red Blockchain. Es una forma específica de prueba diseñada para gestionar la propiedad de activos digitales únicos en la Blockchain.

Los NFTs han ganado una considerable popularidad en el mundo del arte digital y coleccionables digitales, pero también ofrecen numerosas ventajas cuando se utilizan como certificados académicos. Aquí hay algunas razones por las cuales los NFTs son una opción atractiva para certificaciones académicas:

- Únicos e indivisibles: Al igual que en el arte, cada certificado académico emitido como un NFT es único e indivisible. Esto garantiza que no se pueda duplicar ni modificar sin autorización, lo que aumenta su autenticidad y confiabilidad.
- Inmutabilidad: Los NFTs, al estar registrados en la Blockchain, son inmutables, lo que significa que una vez que se emiten, no se pueden modificar ni eliminar. Esto asegura la integridad de los certificados académicos a lo largo del tiempo, evitando la posibilidad de fraude o falsificación.
- Trayectoria y transparencia: Toda la información relacionada con los certificados académicos, como la emisión, la transferencia de propiedad y cualquier cambio de estado, se registra de forma transparente en la Blockchain. Esto proporciona una trayectoria completa y verificable de la autenticidad y legitimidad del certificado.
- Portabilidad y accesibilidad: Los NFTs son compatibles con cualquier plataforma que admita la norma ERC-721, lo que los hace altamente portátiles y accesibles en diferentes entornos y sistemas. Los propietarios de certificados académicos pueden acceder fácilmente a ellos desde cualquier dispositivo con conexión a internet.
- Verificación descentralizada: La información almacenada en la Blockchain es accesible públicamente y se puede verificar de forma descentralizada sin depender de una autoridad central. Esto significa que cualquier persona puede verificar la autenticidad de un certificado académico sin necesidad de intermediarios.
- Propiedad y transferencia seguras: Los NFTs utilizan criptografía avanzada para garantizar la propiedad segura y la transferencia de activos digitales. Cada certificado académico está asociado con una dirección única en la Blockchain, lo que permite a los propietarios transferirlos de manera segura y transparente a otros usuarios.

Para la creación del Contrato inteligente primero abrimos un navegador web y escribimos la url: https://remix.ethereum.org/ para acceder la IDE online de Remix, una vez aquí se crea en la carpeta contracts un contrato inteligente, en nuestro caso se ha creado el contrato UniversityDegree.sol en el que desarrollaremos nuestro contrato inteligente.

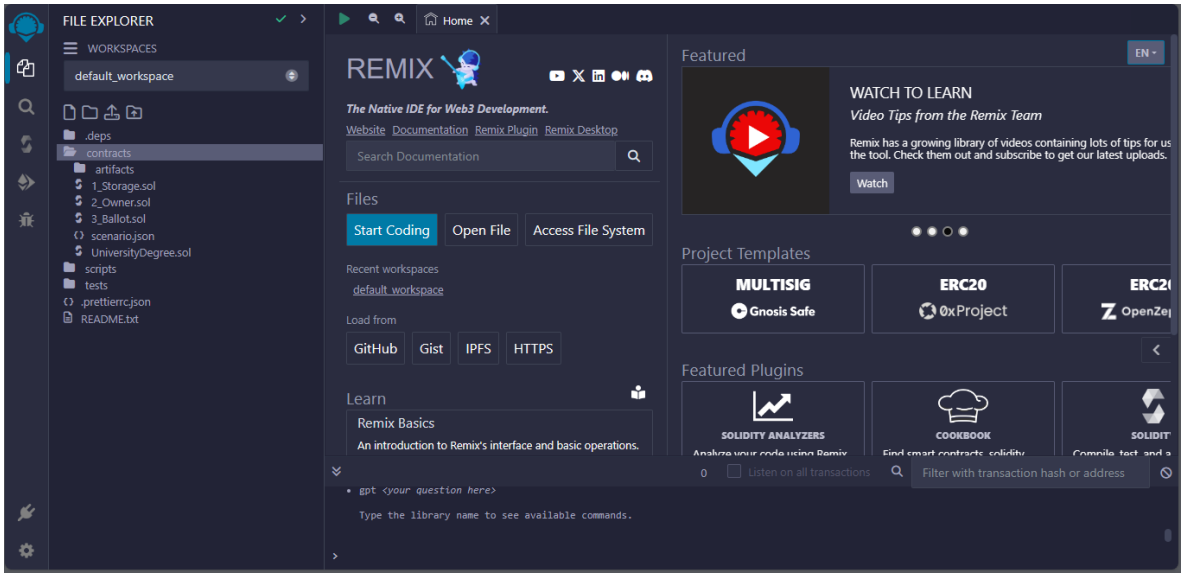


Figura 6. Página principal de Remix.

Al inicio de cada contrato es necesario definir que verison de Solidity es la que se va a usar para que este luego sea compilado, en nuestro caso se ha utilizado 0.8.0 ya que es una versión estable con gran recorrido en la red Ethereum.

```
5   // OpenZeppelin Contracts v4.4.1 (utils/Counters.sol)
6
7   pragma solidity ^0.8.0;
8
```

Figura 7. Código con declaración de versión de Solidity.

Al especificar la versión se incluye un ^ antes del número de la versión para indicar que se puede compilar este contrato con versiones más recientes de la versión, siendo la base de la que se parte la 0.8.0.

Siguiendo la creación del contrato, definimos un constructor para este con algunos elementos básicos.

```
contract UniversityDegree is ERC721URIStorage  {

    address owner;

    using Counters for Counters.Counter;
    Counters.Counter private _tokenIds;

    constructor() ERC721("EDUToken", "Asignatura") {    infinite gas 2283200 gas
        owner = msg.sender;
    }

    mapping(address => string) public personToDegree;
```

Figura 8. Código del constructor del contrato inteligente.

La primera línea declara un nuevo contrato llamado UniversityDegree que hereda de ERC721URIStorage, lo que significa que este contrato hereda todas las funcionalidades de la implementación de la norma ERC-721 que incluye el almacenamiento y la gestión de metadatos de los tokens.

A continuación, se declara una variable de estado owner que almacenará la dirección del propietario del contrato. Y se utiliza la librería Counters que permite mantener un contador interno de los identificadores de los tokens emitidos. La variable _tokenIds será un contador privado de tipo Counters.Counter.

Se declara el constructor del contrato, que se ejecutará una vez al desplegar el contrato. En el constructor, se establece el nombre y el símbolo del token (nombre: "EDUToken" y símbolo: "Asignatura") utilizando la función ERC721. Además, se asigna la dirección que despliega el contrato como el propietario. El símbolo se utilizaría para el nombre de la asignatura, en este caso, se le ha puesto el nombre genérico asignatura y se podría cambiar en un futuro para referirnos a una asignatura específica.

Y se declara un mapeo, en personToDegree donde se socia las direcciones de los usuarios con los URI de los certificados académicos emitidos.

Un detalle importante que aparece en el código es el concepto de gas. El gas es una unidad que mide la cantidad de trabajo computacional requerido para realizar operaciones en la red Ethereum. Es fundamental tanto para la

ejecución de contratos inteligentes como para las transacciones. A la hora de escribir código Remix te hace una estimación del gas que se va a utilizar en una transacción. Hay veces que aparece gas infinito porque se trata de una operación que no se sabe su costo hasta que no esté en tiempo de ejecución. Los usuarios determinan cuánto están dispuestos a pagar por cada unidad de gas en gwei (una denominación de Ether). Un gwei es 1e-9 Ether.

```
function claimDegree(string memory tokenURI)    infinite gas
    public
    returns (uint256)
{
    require(issuedDegrees[msg.sender], "Degree is not issued");

    _tokenIds.increment();

    uint256 newItemId = _tokenIds.current();
    _mint(msg.sender, newItemId);
    _setTokenURI(newItemId, tokenURI);

    personToDegree[msg.sender] = tokenURI;
    issuedDegrees[msg.sender] = false;

    return newItemId;
}
```

Figura 9. Código de la función claimDegree.

En esta parte del código se declara una función pública llamada claimDegree que permite a los usuarios reclamar un certificado académico. Antes de emitir el certificado, verifica si el certificado ha sido previamente emitido para esa dirección. Si es así, incrementa el contador de tokens, emite un nuevo token NFT, asigna el URI del token, actualiza el estado del certificado emitido y retorna el identificador único del nuevo token.

```
function checkDegreeOfPerson(address person) external view returns (string memory) {
    return personToDegree[person];
}
```

Figura 10. Código de la función checkDegreeOfPerson.

A continuación, para que se puede verificar la persona asociada a cada certificado se ha creado una función llamada checkDegreeOfPerson que permite a cualquiera verificar el URI del certificado académico asociado con una dirección dada.

```
modifier onlyOwner() {
    require(msg.sender == owner);
    _;
}

mapping(address => bool) public issuedDegrees;

function issueDegree(address to) onlyOwner external {    27071 gas
    issuedDegrees[to] = true;
}
```

Figura 11. Modificador onlyOwner.

Finalmente, para poder emitir los certificados académicos, primero se ha creado un modificador onlyOwner que hace que asociado a una función esta solo pueda ser efectuada por el propietario del contrato. En la función issueDegree con el modificador onlyOwner permite al propietario del contrato emitir un nuevo certificado académico a una dirección específica.

Además del código mostrado se ha utilizado el código del ERC721 de Oppen Zepeli para poder utilizar las librerías existentes y métodos de los NFT en nuestro EDUToken. Y en este código se ha eliminado la función de SafeTransferFrom para poder hacer que los NFT no puedan ser transferidos entre personas, para mantener la acreditación personal. Aunque hay que resaltar, que en los detalles de los EDUToken aparece el nombre de la persona a la que se le ha transferido inicialmente y la asignatura y Universidad para aumentar la credibilidad de estos, y hacer que aunque sean transferidos no tengan valor ya que esos campos no se pueden modificar.

4.3 IPFS Pinata

IPFS (Sistema de Archivos InterPlanetario) es un protocolo de red peer-to-peer diseñado para crear un sistema de almacenamiento distribuido y descentralizado. Funciona de manera similar a un sistema de archivos tradicional, pero en lugar de almacenar archivos en un servidor centralizado, distribuye los datos en una red de nodos interconectados. Cada archivo en IPFS se identifica mediante un hash único, lo que garantiza su integridad y permite la recuperación eficiente de los datos.

Como se comentó en el capítulo anterior el sistema elegido para este trabajo es Pinata. Pinata es un servicio de alojamiento basado en IPFS que proporciona herramientas y servicios para almacenar y gestionar contenido en la red IPFS de manera fácil y eficiente. Ofrece funciones como la subida de archivos, la gestión de contenido y la generación de enlaces de acceso directo para compartir archivos alojados en IPFS.

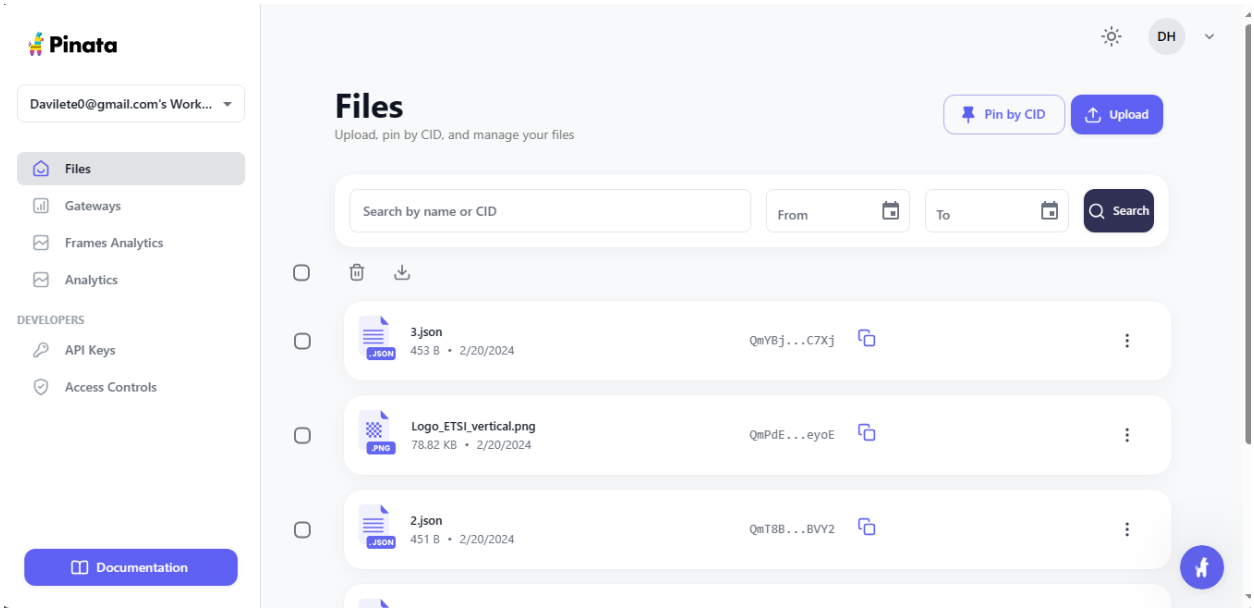


Figura 12. Interfaz de Pinata

Crear una cuenta en Pinata es muy rápido y fácil y se puede tener operativa en un par de minutos. A la hora de la creación se ha escogido el plan gratuito ya que con este podemos suplir de sobra las necesidades de este trabajo. Este plan cuenta con la capacidad de subir hasta 500 archivos, 1 Gb de almacenamiento, 1 Gateway, 10 Gb de ancho de banda y de hasta 10 mil peticiones por mes.

Una vez creada la cuenta se van a subir los archivos necesarios. Los archivos subidos son los JSON con los metadatos del certificado a crear y archivos de imágenes que van a aparecer en el certificado academico. Cada vez que se sube un archivo a la plataforma, se le otorga un CID. Un CID (Content Identifier) en IPFS es un identificador único que se utiliza para referenciar contenido dentro de la red IPFS. Funciona de manera similar a una dirección URL en la web tradicional, pero en lugar de apuntar a una ubicación específica en un servidor, apunta a un recurso específico dentro del sistema de archivos distribuido de IPFS. De esta manera, podremos referenciar a objetos en nuestro contrato inteligente usando el CID de estos.

```
{
    "image":"https://ipfs.io/ipfs/QmPdElnqwZuDSvxf7pbqj6VruDD5c7Nw9GpDayZqNeyoEs",
    "description": "´Nota obtenida en la asignatura X del Grado Y de la Universidad de Sevilla",
    "attributes":   [
        {
            "trait_type": "Nota",
            "value": 7,
            "max-value": 10
        },
        {
            "trait_type": "Grado",
            "value": "Grado en Ingenieria de Y"
        },
        {
            "trait_type": "Nombre Alumno",
            "value": "Fulanito Perez"
        }
    ]
}
```

Figura 13. JSON de un certificado académico.

En el JSON se puede observar que cuenta con 3 campos principales:

- Imagen: que contiene la URL con el CID que apunta a una imagen que previamente hemos subido a Pinata. En este caso es la imagen del logo de la ETSI.
- Descripción: Campo donde se proporciona una pequeña descripción des certificado, en el caso de la figura, se trata de la Nota obtenida en una asignatura de la Universidad de Sevilla.
- Atributos: En este campo se van a especificar los atributos del certificado. En primer lugar se indica el atributo "Nota" con un valor de 7, siendo el máximo 10. El tipo del certificado, en este caso es Grado, con valor el nombre del Grado. Y finalmente el atributo "Nombre Alumno", cuyo valor es el nombre del alumno, en el ejemplo Fulanito Pérez.

Figura 14. Logo de la ETSI a la que apunta la url del JSON anterior.[8]

4.4 MetaMask

MetaMask es la herramienta que se va a utilizar como cartera para hacer las transacciones en la red de Ethereum. MetaMask es una extensión de navegador y una aplicación móvil que permite a los usuarios interactuar con la Blockchain de Ethereum. Funciona como una billetera digital para almacenar y gestionar criptomonedas y tokens ERC-20, así como para ejecutar y gestionar contratos inteligentes. MetaMask facilita el acceso a aplicaciones descentralizadas (dApps) directamente desde el navegador.

Figura 15. Logo de Metamask. [9]

[8] Imagen tomada de etsi.us.es
[9] Imagen tomada de https://safeswap.io/wallet/metamask/

Para instalar y configurar MetaMask, se han seguido estos pasos:

- Instalación: Desde la tienda de extensiones del navegador (Chrome, Firefox, Brave, Edge) se busca "MetaMask" y se procede a descargar e instalar la extensión.
- Creación de Cuenta: Una vez instalada, se abre MetaMask y selecciona "Crear una billetera". A continuación, se han seguido las instrucciones para crear una contraseña segura.
- Frase de Recuperación: MetaMask proporciona una frase de recuperación de 12 palabras. Es fundamental guardar esta frase en un lugar seguro, ya que es la única manera de recuperar tu billetera si pierdes el acceso.
- Configuración: una vez completa la configuración inicial la billetera estará lista para usarse.

Al abrir MetaMask se puede ver tu dirección de Ethereum y empezar a recibir y enviar criptomonedas.

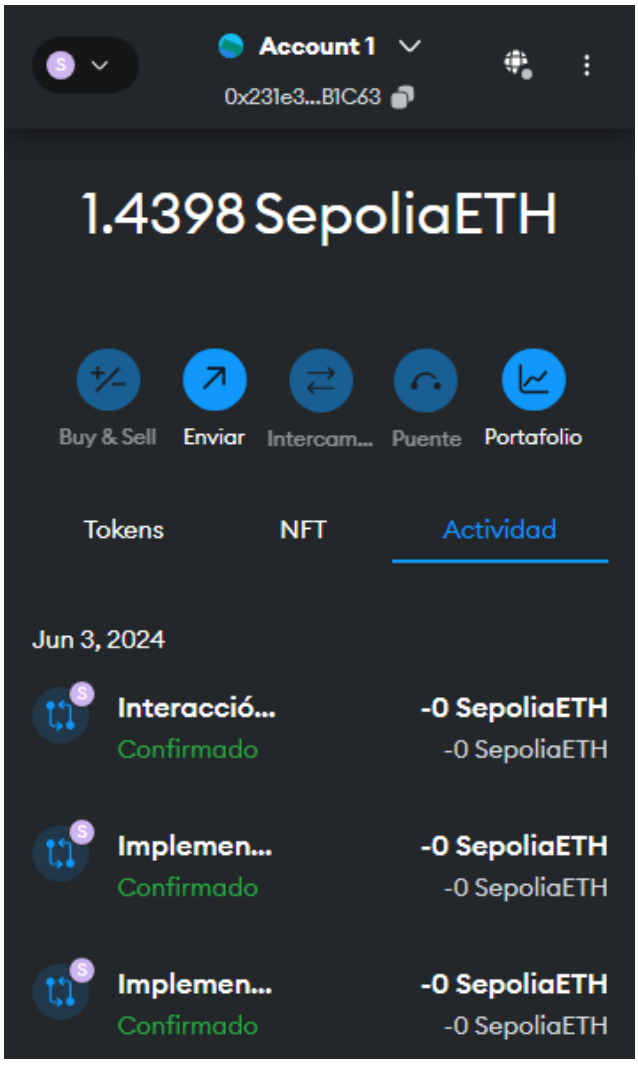


Figura 16. Wallet de una cuenta en Metamask.

Para simular el comportamiento de acreditación se han creado dos cuentas en MetaMask, la primer correspondiente con el profesor, cuya dirección es 0x231e329b7074a3cb72D165A7000CE7b0279B1C63, y la

segunda, correspondiente al alumno cuya dirección es 0x8Ad8d454cA5F983699c42c725e73e6A5d8042Dc2.

Como los tokens y las transacciones en la red Ethereum son muy costosos y este proyecto solo se limita a diseñar y caracterizar un sistema de acreditación para su evaluación, todas las transacciones y despliegues de contratos se va a realizar en la red de pruebas Sepolia.

Las redes de prueba, como Sepolia, son entornos de Blockchain que replican las características de la red principal (mainnet) de Ethereum pero no utilizan criptomonedas reales. Estas redes permiten a desarrolladores y usuarios probar aplicaciones, contratos inteligentes y transacciones sin el riesgo de perder dinero real.

Para conseguir tokens en esta red de prueba se ha utilizado un facuer. Un faucet es un servicio que distribuye pequeñas cantidades de criptomonedas de prueba gratuitamente. En el contexto de redes de prueba como Sepolia, los faucets proporcionan tokens de prueba que los desarrolladores pueden utilizar para realizar pruebas en sus aplicaciones y contratos inteligentes sin costo alguno. Para obtener tokens de prueba, los usuarios visitan un sitio web de faucet, ingresan su dirección de la red de prueba y solicitan los tokens necesarios. Estas paginas web suelen dar pequeñas cantidades de tokens de manera gratuita cada cierto tiempo o haciendo actividades como registrándose con una dirección de correo o similares.

Para la prueba del sistema se han conseguido unos pocos tokens, tanto para la cuenta del profesor como la del alumno, para las actividades de desplegar el contrato inteligente, crear el certificado o para minarlo.

4.5 Pasos para crear el EduToken

1-compilar Smart contract

El primer paso de todos sería entrar en el IDE Remix desde el navegador y compilar el contrato inteligente UniversityDegree.sol. Esto lo haríamos desde la pestaña solidity compiler y haciendo click en el botón de compile UiversityDegree.sol

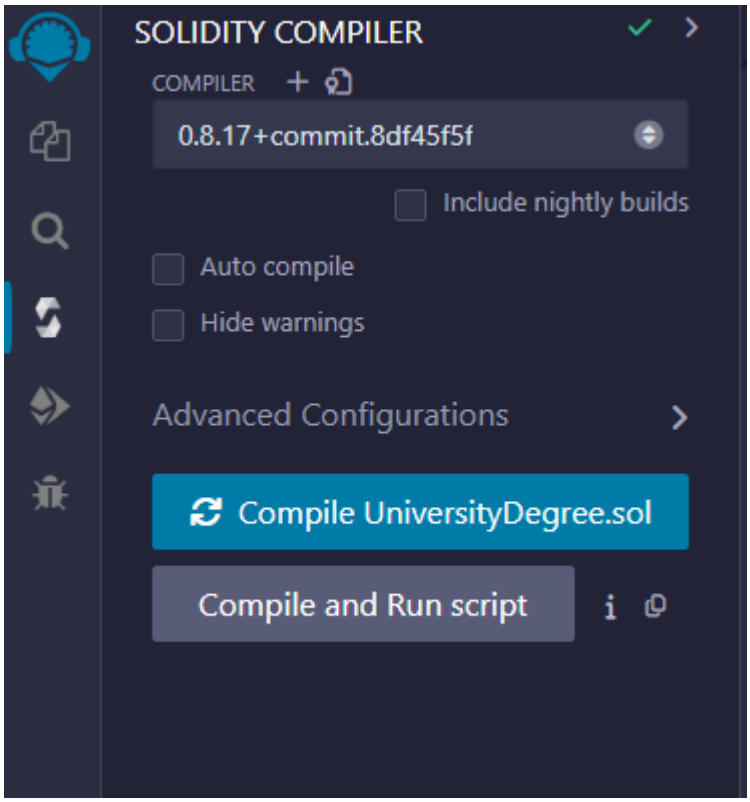


Figura 17. Compilación Smart Contract.

2-una vez compilado el contrato inteligente sin errores, en la pestaña deploy and run transactions, elegimos injected provider Metamask. A continuación nos saltará una ventana de MetaMask para confirmar que queremos conectarnos con Remix y seleccionamos la cuenta del profesor y aceptamos.

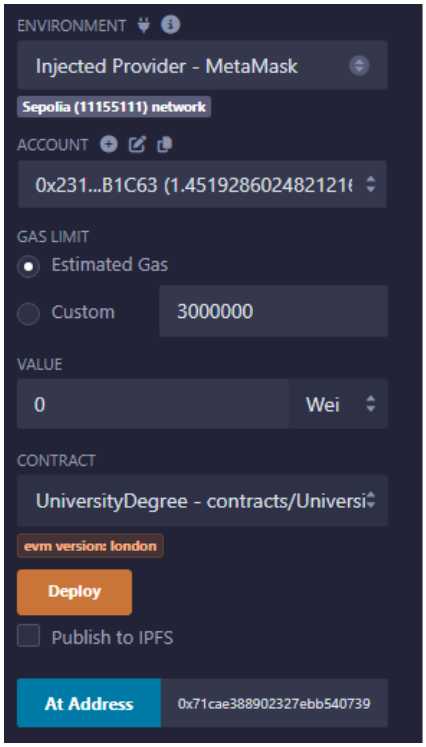


Figura 18. Despliegue Smart Contract.

Podemos ver como tenemos seleccionada la cuenta del profesor, la que empieza por 0x23. Hacemos click en

Deploy y se nos generará automáticamente una dirección en la que se creará el Smart Contract. En este caso 0x71cae388902327ebb540739.

Nuevamente, aparece una ventana de MetaMask, pero esta vez para confirmar la transacción.

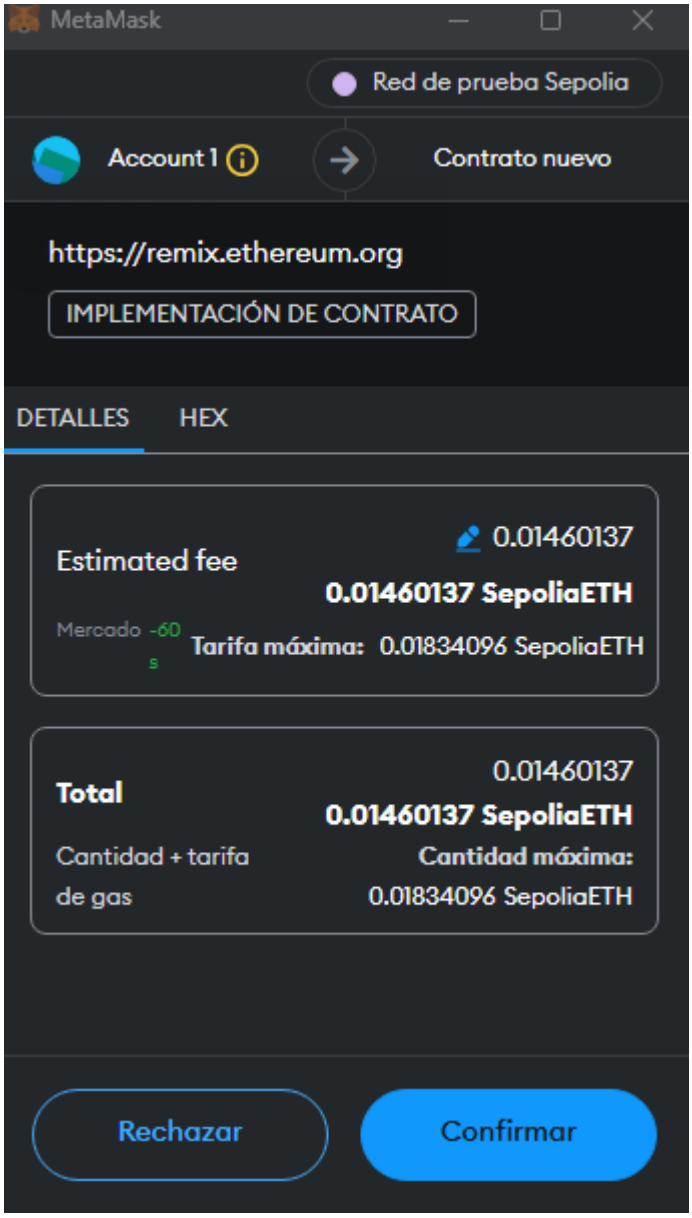


Figura 19. Ventana de confirmación de Transacción.

Desplegar un contrato inteligente consume ethers, en este caso es de unos 0.01460137 SepoliaETH.

Una vez confirmamos la transacción, al pasar unos segundos en la consola de Remix aparece transacción exitosa:

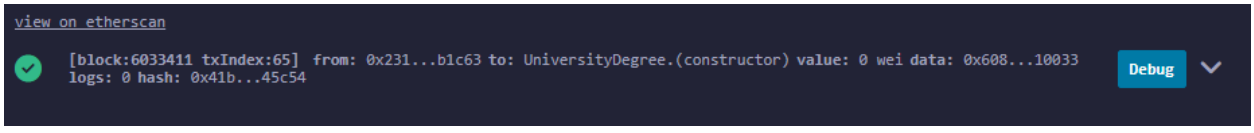


Figura 20. Mensaje de transacción exitosa.

Aquí podemos ver como se ha creado el Smart Contract de UniversityDegree.sol, el bloque en el que está y la cuenta que lo creó. Además, aparece la opción de vie won etherscan para ver la transacción con más detalles.

Etherscan es un explorador de Blockchain para la red Ethereum. Permite a los usuarios buscar y verificar transacciones, contratos inteligentes, direcciones de cartera y otros datos en la Blockchain de Ethereum. Etherscan ofrece una interfaz amigable para visualizar información detallada sobre el estado y el historial de transacciones, saldos de cuentas, bloques minados, y otros eventos en la red. Es una herramienta esencial para desarrolladores, inversores y usuarios que desean monitorear y analizar la actividad en la red Ethereum.

Desde esta herramienta en el navegador podemos ver todos los datos de la transacción.

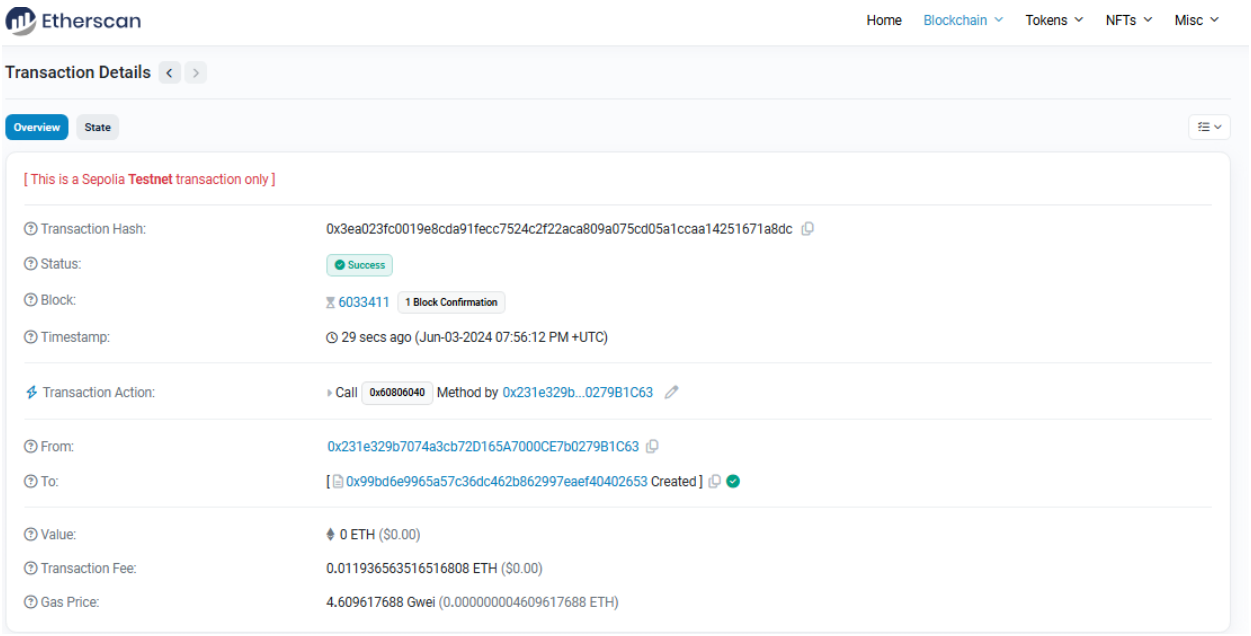


Figura 21. Transacción en Etherscan.

Aparecen datos como eh Hash, el estado de la transacción, exitosa en este caso, cuándo se realizó, quién la realizó y la dirección del contrato: 0x99bd6e9965a57c36dc462b862997eaef40402653

Volviendo de nuevo a Remix, ahora aparecen todos los métodos del contrato en Remix:

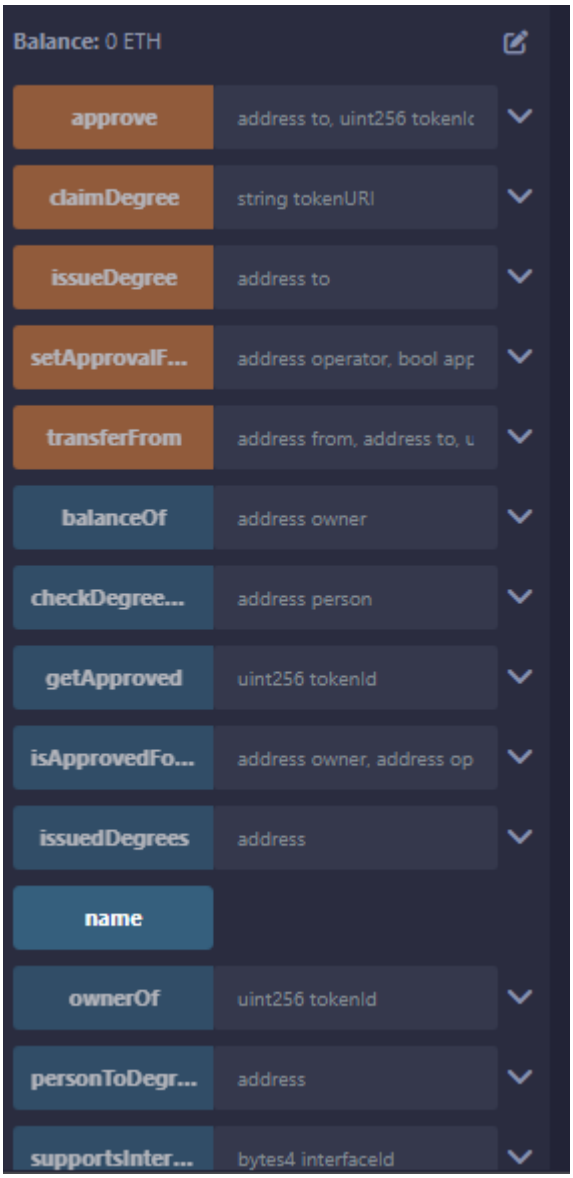


Figura 22. Lista de funciones del contrato inteligente.

Utilizando la opción de issueDegree ponemos la dirección de la segunda cuenta, la del alumno (0x8Ad8d454cA5F983699c42c725e73e6A5d8042Dc2) para emitir su certificado.

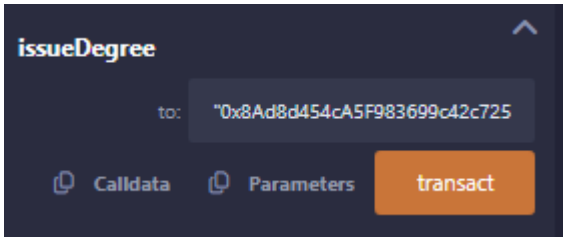


Figura 23. Función issueDegree.

Y nuevamente, aparece la ventana para confirmar transacción de MetaMask.

Figura 24. Ventana de confirmación de transacción de MetaMask.

En este caso la tarifa es solo de 0.0002056 SepoliaETH, ya que el método de issueDegree es mucho menos costoso computacionalmente que el de desplegar el contrato en la red.

En la consola de Remix, aparece nuevamente la transacción exitosa:

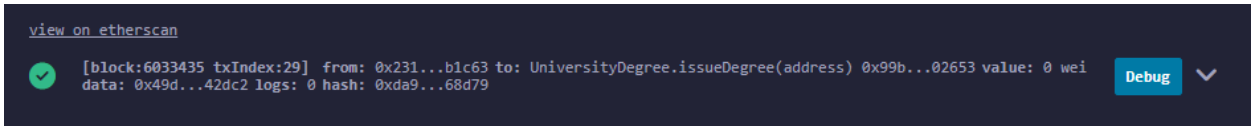


Figura 25. Mensaje de transacción 2 exitosa.

Haciendo uso de la herramienta de Etherscan nuevamente, podemos ver todos los detalles de la transacción.

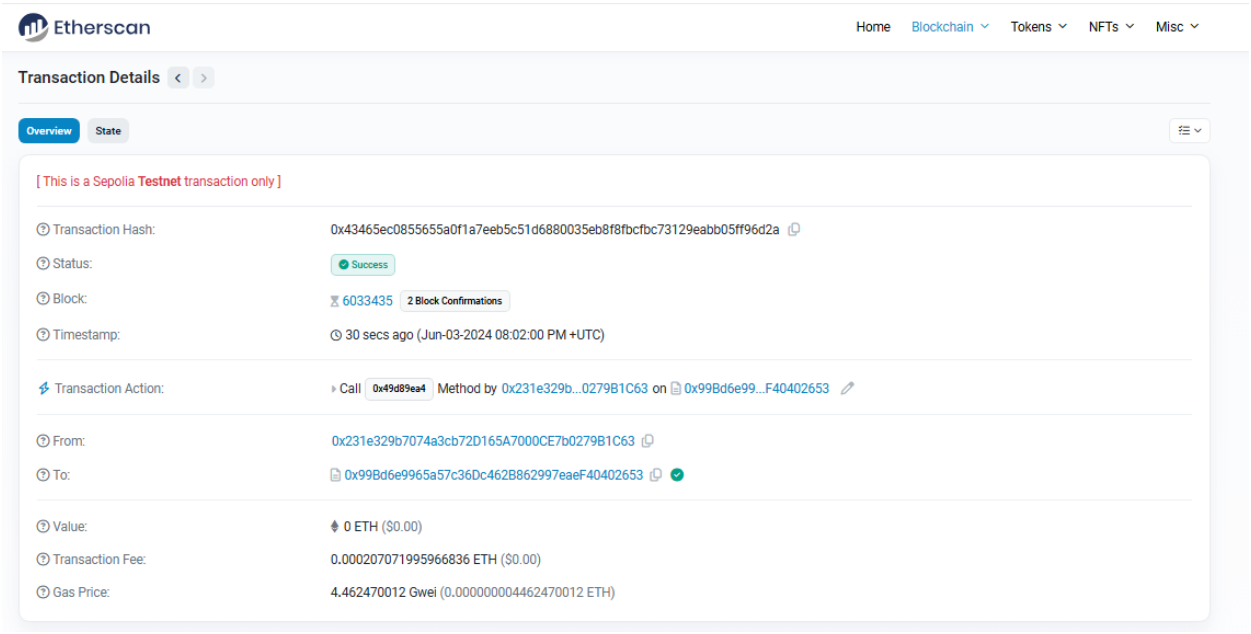


Figura 26. Transacción 2 en Etherscan.

A continuación, el profesor sube a Pinata, el JSON con los datos del alumno y le envía al alumno el CID de su certificado.

Ahora desde la cuenta 2, es decir la del alumno, utilizamos la opción de claimDegree, poniendo en el campo la url del CID del JSON de su certificado de Pinata que el profesor le proporcionará.

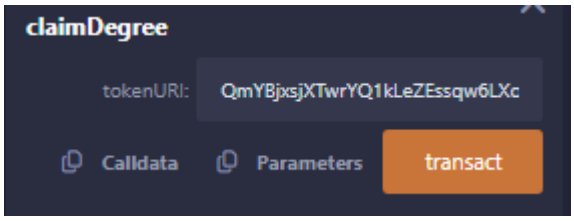


Figura 27. Función claimDegree.

De nuevo, aceptamos la transacción en MetaMask.

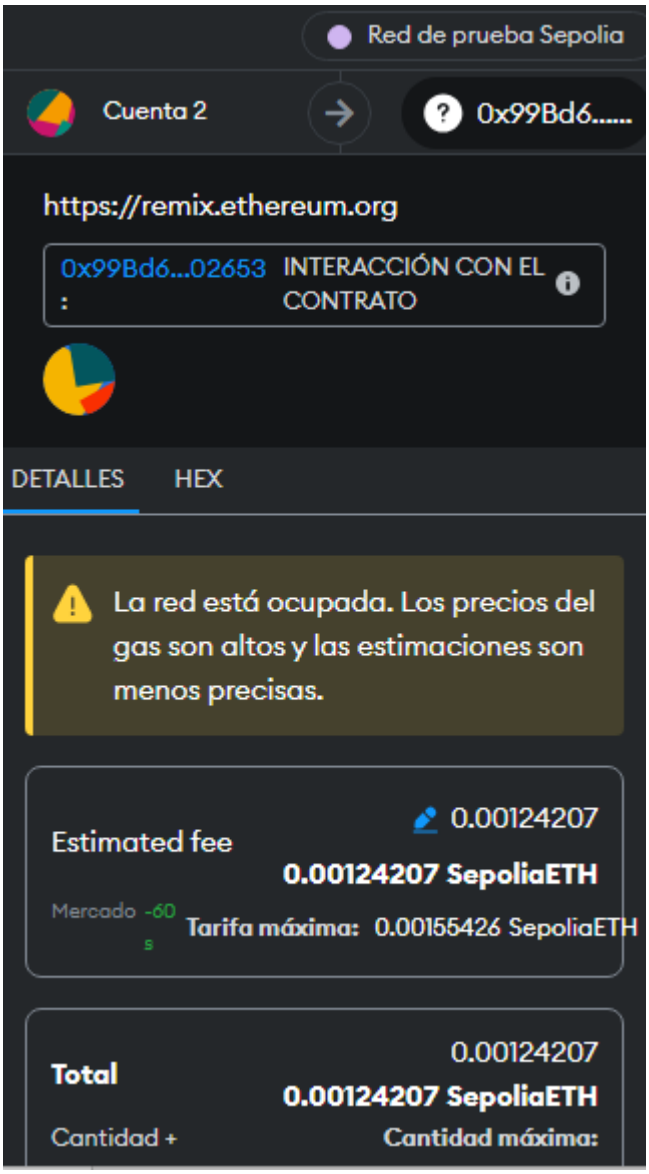


Figura 28. Confirmación transacción 3 en MetaMask.

Y la transacción vuelve a ser exitosa.

Figura 29. Mensaje de transacción 3 exitosa.

Siendo la salida en Etherscan la siguiente:

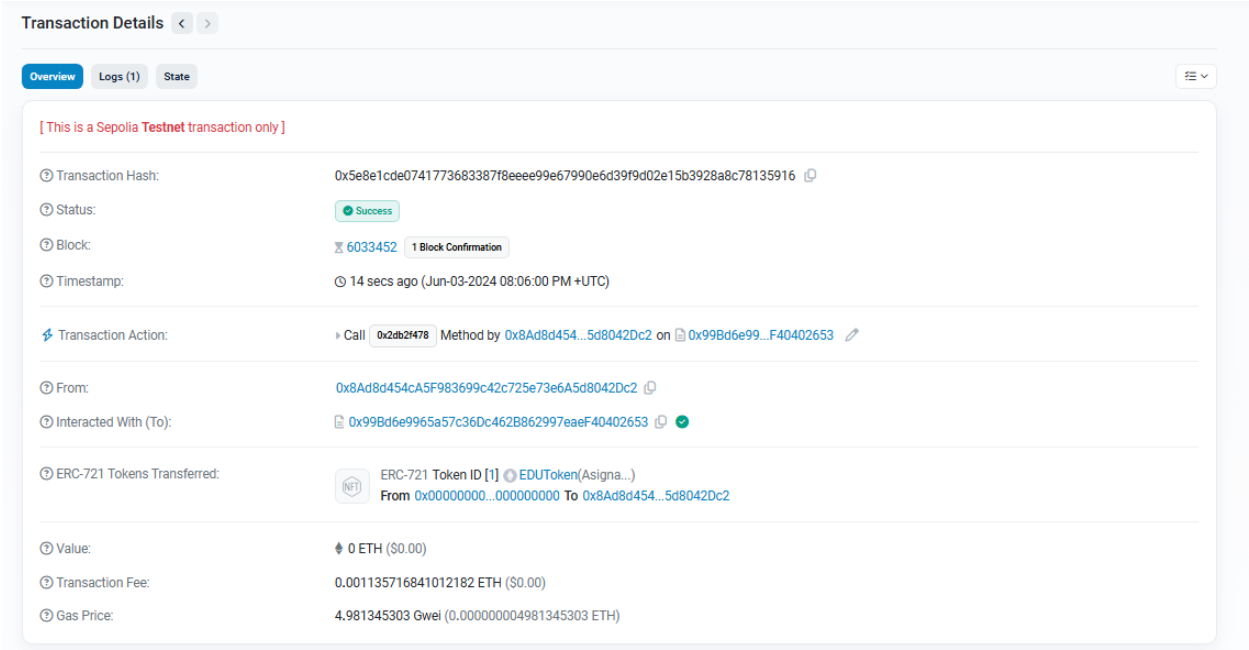


Figura 30. Transacción 3 en Etherscan.

Ahora aparece que estamos usando un ERC-721(NFT) con el token que estamos usando EDUToken

Una vez hecho esto ya estaría acabado el proceso de certificación digital. En testnets.opensea.io podemos visualizar los tokens del alumno:

OpenSea es un mercado en línea descentralizado para la compra, venta y comercio de tokens no fungibles (NFTs). Los NFTs son activos digitales únicos que pueden representar una variedad de objetos digitales, como arte, coleccionables, música, y objetos dentro de videojuegos. OpenSea permite a los usuarios crear, descubrir, y negociar estos activos usando la tecnología Blockchain, principalmente sobre la red Ethereum.

En OpenSea, los usuarios pueden:

- Explorar y comprar NFTs: Los usuarios pueden navegar por una amplia variedad de NFTs listados en la plataforma y comprarlos utilizando criptomonedas, principalmente Ether (ETH).
- Crear y vender NFTs: Los artistas y creadores pueden acuñar sus propios NFTs y venderlos en el mercado, estableciendo precios fijos o subastas.
- Interactuar con colecciones: OpenSea alberga diversas colecciones de NFTs, facilitando la organización y la gestión de estos activos digitales.

OpenSea se ha convertido en uno de los mercados más grandes y populares para NFTs debido a su amplia gama de categorías y su facilidad de uso.

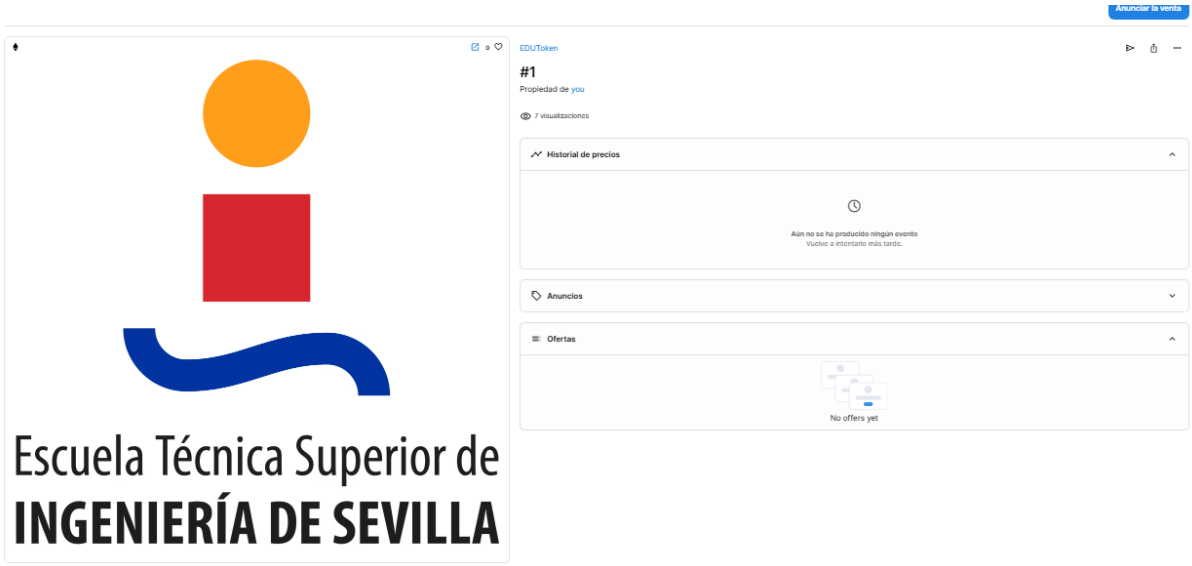


Figura 31. NFT del certificado educativo en OpenSea.

Aquí podemos ver los detalles del token como, la dirección que lo creó, la del profesor, una breve descripción, unos atributos, la nota del alumno en la asignatura. Y en más detalles podemos ver la dirección del contrato, el ID, el estándar y la cadena en la que está entre otras cosas.

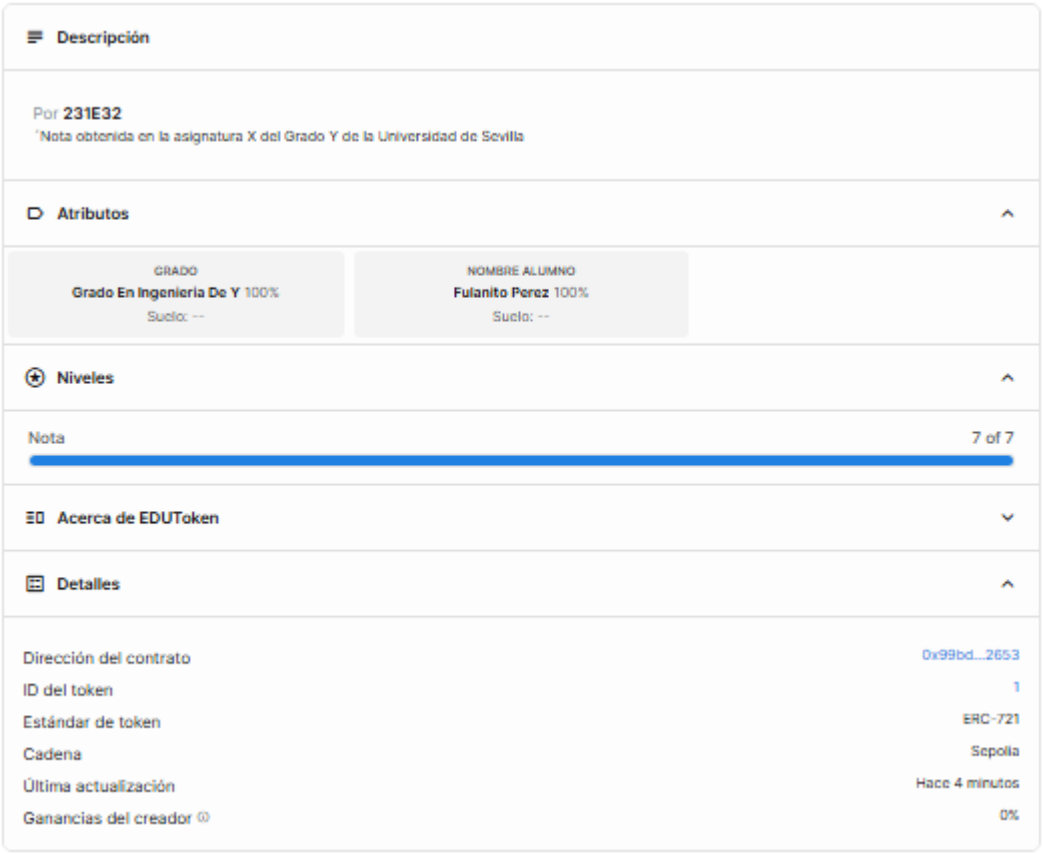


Figura 32. Detalles del NFT.

Algunas funciones adicionales que nos permite realizar son:

En el Smart contract podemos usar las opciones checkDegreeofPerson e introduciendo la dirección de la cuenta

2, devuelve el CID del certificado.

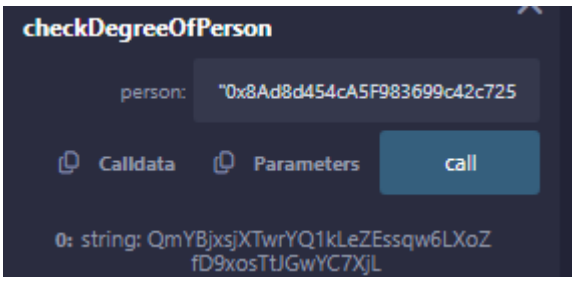
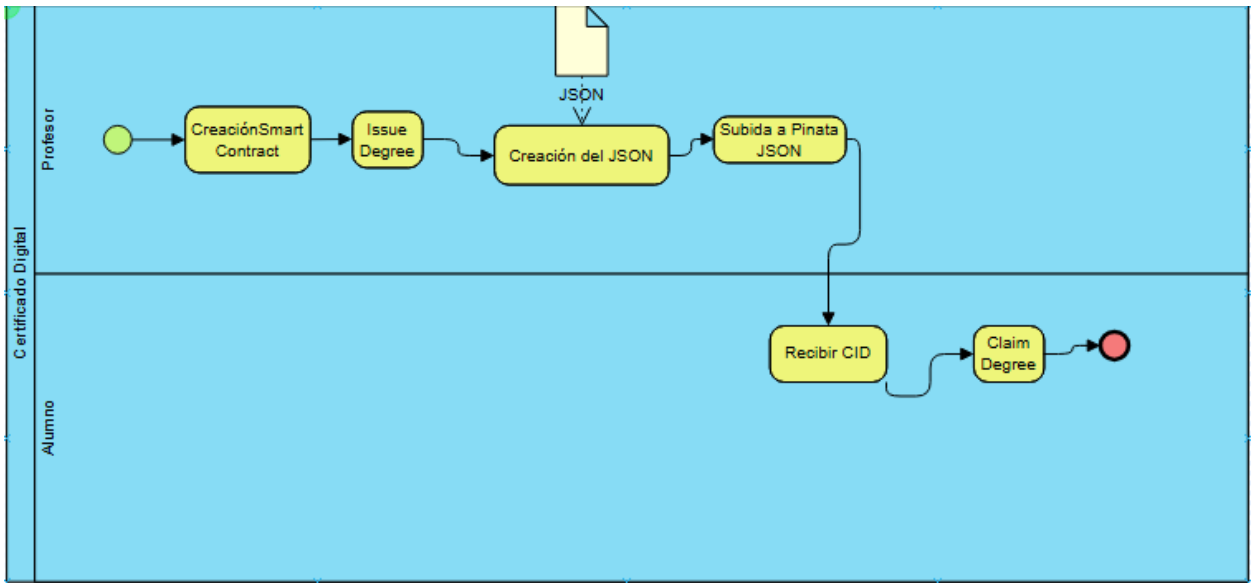


Figura 33. Función checkDegreeOfPerson.

4.6 Diagrama BPMN

A continuacion se detalla un diagrma BPMN con los pasos que se realizan durante todo el proceso de acreditación.



Un diagrama BPMN (Business Process Model and Notation) es una representación gráfica de un proceso empresarial. Utiliza símbolos estandarizados para visualizar las actividades, flujos de trabajo, decisiones y eventos que componen un proceso. Estos diagramas son utilizados para modelar, analizar y mejorar los procesos empresariales, facilitando la comunicación entre los diferentes miembros de una organización y ayudando a identificar áreas de mejora y eficiencia.

En el diagrama BPMN se presentan las acciones llevadas a cabo tanto por el profesor de la asignatura, quien acredita al alumno, como por el propio alumno que ha superado la asignatura y busca ser acreditado. Se detallan los intercambios de mensajes y los pasos seguidos, que son los siguientes:

1. Creación del contrato inteligente: Un profesor crea un contrato inteligente en la Blockchain que define las reglas y condiciones para la emisión de calificaciones a los estudiantes. Este contrato inteligente

actúa como un intermediario confiable que garantiza la ejecución automatizada y segura de las condiciones acordadas.

2. Generación del token para el estudiante: Cuando el profesor califica a un estudiante en una asignatura, genera un archivo JSON que contiene información relevante, como la nota, el nombre del alumno, la asignatura y otros detalles pertinentes. Este archivo JSON actúa como el certificado digital de la calificación otorgada.

3. Subida del archivo JSON a IPFS (sistema de archivos interplanetario): El archivo JSON generado se sube a IPFS, un sistema de archivos descentralizado que utiliza la tecnología Blockchain para almacenar y distribuir contenido de manera descentralizada. IPFS asigna un identificador único al archivo, conocido como CID (Content Identifier), que permite acceder al archivo desde cualquier lugar de la red IPFS.

4. Almacenamiento del CID en Pinata: Pinata es un servicio que facilita el almacenamiento y la gestión de archivos en IPFS de manera conveniente. El profesor utiliza Pinata para subir el archivo JSON a IPFS y obtiene un token único de Pinata que representa el archivo subido.

5. Entrega del CID al estudiante: Una vez que el archivo JSON está almacenado en IPFS y se ha obtenido el token de Pinata, el profesor entrega este token al estudiante. El token actúa como prueba criptográfica de la existencia e integridad del certificado de calificación del estudiante en IPFS.

6. Reclamación del certificado por parte del estudiante: El estudiante puede reclamar su certificado de calificación utilizando el token recibido del profesor. Al presentar este token, el estudiante puede acceder al archivo JSON almacenado en IPFS y verificar la autenticidad de su calificación.

5 RESULTADOS Y CONCLUSIONES

5.1 Resultados del Sistema Desarrollado

El sistema de certificación académica implementado en la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla utiliza tecnología Blockchain y NFTs para mejorar la gestión y verificación de credenciales académicas. A continuación, se desarrollan los resultados obtenidos, destacando sus beneficios, facilidad de uso y posibles complicaciones.

Los beneficios del sistema son:

- Seguridad Mejorada: Inmutabilidad de los Datos: Los registros académicos almacenados en la Blockchain no pueden ser alterados una vez creados, asegurando que las calificaciones y certificaciones sean auténticas y no manipulables.
- Prevención de Fraudes: La transparencia de la Blockchain permite a cualquier entidad verificar la autenticidad de un certificado, reduciendo el riesgo de falsificaciones.
- Transparencia y Confianza: Todas las transacciones y registros son accesibles públicamente en la Blockchain, permitiendo una transparencia completa y garantizando la integridad de los datos.
- Confianza en las Credenciales: Tanto empleadores como instituciones educativas pueden verificar de manera independiente y confiable las credenciales de los estudiantes, fortaleciendo la credibilidad del curriculum vitae de los graduados.
- Reducción de Burocracia: La eliminación de procesos manuales y documentos en papel agiliza la emisión y verificación de certificados, eliminando barreras burocráticas.
- Empoderamiento de los Estudiantes, propiedad y control: Los estudiantes tienen control total sobre sus certificados digitales, pudiendo compartirlos fácilmente con potenciales empleadores o instituciones educativas.
- Acceso Global: Los certificados almacenados en IPFS son accesibles desde cualquier lugar del mundo, facilitando la movilidad académica y profesional.
- Facilidad de Uso
- Herramientas de Desarrollo: El uso de herramientas como Remix e IDE facilita la creación y prueba de contratos inteligentes, haciendo el desarrollo accesible incluso para usuarios con conocimientos técnicos básicos.
- Gestión de Archivos en IPFS: Servicios como Pinata simplifican la subida y gestión de archivos en IPFS, proporcionando una interfaz fácil de usar para profesores y estudiantes.
- Verificación Instantánea: Los certificados pueden ser verificados instantáneamente por cualquier parte interesada utilizando el CID, eliminando la necesidad de procedimientos de verificación complicados.

Posibles Complicaciones

- Costos de Transacción: Gas Fees en Ethereum: Las tarifas de transacción en la red Ethereum pueden variar y, en momentos de alta demanda, pueden ser costosas. Esto podría representar un desafío para la escalabilidad del sistema.
- Curva de Aprendizaje: Conocimientos Técnicos: La implementación y mantenimiento del sistema requiere conocimientos técnicos específicos en Blockchain y contratos inteligentes, lo cual puede ser una barrera inicial para algunas instituciones.
- Seguridad de Datos en IPFS:Aunque IPFS es un sistema descentralizado, la disponibilidad de los archivos depende de la cantidad de nodos que los almacenen. Es crucial asegurar que los archivos importantes se mantengan disponibles y no sean comprometidos.
- Regulación y Cumplimiento: Marcos Legales: La adopción de tecnologías Blockchain en el ámbito académico debe alinearse con las regulaciones locales e internacionales sobre protección de datos y certificación académica, lo cual puede requerir ajustes y revisiones legales.
- Baja escalabilidad: el sistema actual no escalaría muy bien para poder tener en cuenta marcos más amplios como grados enteros, o titulaciones.

5.1 Conclusiones y trabajo futuro

La tecnología Blockchain se encuentra apenas en sus primeras etapas de desarrollo. Este desarrollo no es sencillo puesto que la infraestructura que necesita para su implantación no se encuentra disponible y requiere de una disrupción total del modelo económico y tecnológico existente. La situación descrita es perfectamente extrapolable al ámbito académico, en el que podemos concluir de los estudios y primeros casos de uso publicados que ya se han puesto los primeros bloques para la construcción y desarrollo de la tecnología. En este trabajo se ha discutido y explorado el impacto que puede tener la tecnología Blockchain en el sector educativo, presentándose un ejemplo de implementación que permite certificar notas de alumnos de una asignatura impartida en la Escuela Técnica Superior de Ingenieros de la Universidad de Sevilla. Se han analizado las capacidades y limitaciones actuales de la tecnología, así como las posibilidades que ésta ofrece de cara a un futuro cercano, discutiéndose los principales desafíos que se han de abordar, sobre todo relacionados con la privacidad, seguridad, confianza y confiabilidad, verificación y validación de los datos, así como con la gestión, adopción, conocimiento técnico, regulación y coste de la tecnología.

Gracias al sistema desarrollado he podido aprender acerca de un lenguaje nuevo como es Solidity para la creación del contrato inteligente, y he podido aprender mucho de la tecnología de los contratos inteligentes y de las Dapps. He podido reconocer las fortalezas y utilidades que presenta la tecnología y ver como se pueden aplicar en un contexto real practico.

Finalmente, como trabajo a futuro, se podría extender el sistema de acreditación para evaluar no solo la superación de una asignatura, si no también, la acreditación de títulos enteros, excedidos por distintas entidades o el de evaluación de conocimientos como pueden ser cursos o formaciones complementarias. Además, otra rama en la que se puede mejorar es el uso de cadenas de bloques privadas como puede ser HyperLedger para poder realizar un sistema que se pueda escalar de manera fácil, siendo el coste y la complejidad de este el principal tema a abordar.

REFERENCIAS

[1] M. Belotti, N. Božić, G. Pujolle, S. Secci, "A Vademecum on Blockchain Technologies: When, Which, and How," IEEE Communications Surveys & Tutorials, vol. 21, no. 4, pp. 3796-3838, 2019.

[2] Anjee Gorkhali , Ling Li, Asim Shrestha, "Blockchain: a literature review," Journal of Management Analytics, vol. 7, no. 3, pp. 321-343, 2020.

[3] A. El Koshiry, E. Eliwa, T.A. El-Hafeez, M.Y. Shams, "Unlocking the power of blockchain in education: An overview of innovations and outcomes," Blockchain: Research and Applications, vol. 4, Iss. 4, pp. 1-19, 2023.

[4] M.A. Haque, S. Haque, S. Zeba, et al., "Sustainable and efficient E-learning internet of things system through blockchain technology," E-Learning and Digital Media, pp. 1-20, 2023.

[5] R. Bhatia, N.K. Bhasin, "A study of the new role of blockchain in the Indian education system," Int. J. e-Collaboration, vol. 19, Iss. 1, pp. 1-19, 2023.

[6] F. Vidal, F. Gouveia and C. Soares, "Analysis of Blockchain Technology for Higher Education," 2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), Guilin, China, pp. 28-33, 2019.

[7] S.P. Dash, "An Introduction to Blockchain Technology: Recent Trends," Panda, S.K., Mishra, V., Dash, S.P., Pani, A.K. (eds) Recent Advances in Blockchain Technology. Intelligent Systems Reference Library, vol. 237, Springer, Cham, 2023.

[8] Sirus Mashouf Mohsenin, "Blockchain: contratación y jurisdicción," Revista Aranzadi de Derecho y Nuevas Tecnologías, Nº. 60/2022 (BIB 2022\3537).

[9] Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales.

[10] Agencia Española de Protección de Datos (AEPD), "Introducción al HASH como técnica de seudonimización de datos personales," https://www.aepd.es/documento/estudio-hash-anonimidad.pdf, 2019.

[11] M. Costeja González, STJUE (GS) de 13 de mayo de 2014, Asunto C.131/12, Google vs. Agencia Española de Protección de Datos.

[12] https://github.com/dappuniversity/Soul-Bound-token

[13] Richter Vidal, F., Gouveia, F., & Soares, C. Analysis of Blockchain Technology for Higher Education. (2019, 1 octubre). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8945824

[14] Turkanovic, M., Holbl, M., Kosic, K., Hericko, M., & Kamisalic, A. (2018). EduCTX: a Blockchain-Based higher education credit platform. IEEE Access, 6, 5112-5127. https://doi.org/10.1109/access.2018.2789929

[15] Mohammad, A., & Vargas, S. (2022). Challenges of Using Blockchain in the Education Sector: A Literature Review. Applied Sciences, 12(13), 6380. https://doi.org/10.3390/app12136380

[16] Upadhyay, N. (2020). Demystifying blockchain: A critical analysis of challenges, applications and opportunities. International Journal Of Information Management, 54, 102120. https://doi.org/10.1016/j.ijinfomgt.2020.102120

[17] Vidal, F. R., Gouveia, F., & Soares, C. (2020). Revocation Mechanisms for Academic Certificates Stored on a Blockchain. IEEE XPlore. https://doi.org/10.23919/cisti49556.2020.9141088

[18] https://www.geeksforgeeks.org/introduction-to-solidity/

[19] https://docs.soliditylang.org

[20] https://solidity-by-example.org/

[21] https://ethereum.org/en/developers/docs/consensus-mechanisms/pow/

[22] https://bitcoin.org/bitcoin.pdf

[23] https://www.investopedia.com/terms/p/proof-work.asp

[24] https://www.ibm.com/blockchain/what-is-blockchain/consensus-models

[25] https://www.statista.com/statistics/621207/worldwide-blockchain-startup-financing-history/

[26] https://ethereum.github.io/yellowpaper/paper.pdf

ANEXO

En este anexo se muestra el código empleado para el contrato inteligente, además de los JSON utilizados para la acreditación de ejemplo presentada en el trabajo.

```
// SPDX-License-Identifier: MIT
// File: @openzeppelin/contracts/utils/Counters.sol


// OpenZeppelin Contracts v4.4.1 (utils/Counters.sol)

pragma solidity ^0.8.0;



/**
 * @title Counters
 * @author Matt Condon (@shrugs)
 * @dev Provides counters that can only be incremented, decremented or reset. This can be used e.g. to track the number
 * of elements in a mapping, issuing ERC721 ids, or counting request ids.
 *
 * Include with `using Counters for Counters.Counter;`
 */
library Counters {
    struct Counter {
        // This variable should never be directly accessed by users of the library: interactions must be restricted to
        // the library's function. As of Solidity v0.5.2, this cannot be enforced, though there is a proposal to add
        // this feature: see https://github.com/ethereum/solidity/issues/4637
        uint256 _value; // default: 0
    }

    function current(Counter storage counter) internal view returns (uint256) {
        return counter._value;
```

```
    }

    function increment(Counter storage counter) internal {
        unchecked {
            counter._value += 1;
        }
    }

    function decrement(Counter storage counter) internal {
        uint256 value = counter._value;
        require(value > 0, "Counter: decrement overflow");
        unchecked {
            counter._value = value - 1;
        }
    }

    function reset(Counter storage counter) internal {
        counter._value = 0;
    }
}

// File: @openzeppelin/contracts/utils/Strings.sol


// OpenZeppelin Contracts v4.4.1 (utils/Strings.sol)


/**
 * @dev String operations.
 */
library Strings {
    bytes16 private constant _HEX_SYMBOLS = "0123456789abcdef";

    /**
     * @dev Converts a `uint256` to its ASCII `string` decimal representation.
     */
    function toString(uint256 value) internal pure returns (string memory) {
        // Inspired by OraclizeAPI's implementation - MIT licence
```

```
    // https://github.com/oraclize/ethereum-api/blob/b42146b063c7d6ee1358846c198246239e9360e8/oraclizeAPI_0.4.25.sol

    if (value == 0) {
        return "0";
    }
    uint256 temp = value;
    uint256 digits;
    while (temp != 0) {
        digits++;
        temp /= 10;
    }
    bytes memory buffer = new bytes(digits);
    while (value != 0) {
        digits -= 1;
        buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
        value /= 10;
    }
    return string(buffer);
}

/**
 * @dev Converts a `uint256` to its ASCII `string` hexadecimal representation.
 */
function toHexString(uint256 value) internal pure returns (string memory) {
    if (value == 0) {
        return "0x00";
    }
    uint256 temp = value;
    uint256 length = 0;
    while (temp != 0) {
        length++;
        temp >>= 8;
    }
    return toHexString(value, length);
}

/**
```

```
     * @dev Converts a `uint256` to its ASCII `string` hexadecimal representation with fixed length.
     */
    function toHexString(uint256 value, uint256 length) internal pure returns (string memory) {
        bytes memory buffer = new bytes(2 * length + 2);
        buffer[0] = "0";
        buffer[1] = "x";
        for (uint256 i = 2 * length + 1; i > 1; --i) {
            buffer[i] = _HEX_SYMBOLS[value & 0xf];
            value >>= 4;
        }
        require(value == 0, "Strings: hex length insufficient");
        return string(buffer);
    }
}

// File: @openzeppelin/contracts/utils/Context.sol


// OpenZeppelin Contracts v4.4.1 (utils/Context.sol)



/**
 * @dev Provides information about the current execution context, including the
 * sender of the transaction and its data. While these are generally available
 * via msg.sender and msg.data, they should not be accessed in such a direct
 * manner, since when dealing with meta-transactions the account sending and
 * paying for execution may not be the actual sender (as far as an application
 * is concerned).
 *
 * This contract is only required for intermediate, library-like contracts.
 */
abstract contract Context {
    function _msgSender() internal view virtual returns (address) {
        return msg.sender;
    }

    function _msgData() internal view virtual returns (bytes calldata) {
```

```
        return msg.data;
    }
}

// File: @openzeppelin/contracts/utils/Address.sol


// OpenZeppelin Contracts (last updated v4.5.0) (utils/Address.sol)


/**
 * @dev Collection of functions related to the address type
 */
library Address {
    /**
     * @dev Returns true if `account` is a contract.
     *
     * [IMPORTANT]
     * ====
     * It is unsafe to assume that an address for which this function returns
     * false is an externally-owned account (EOA) and not a contract.
     *
     * Among others, `isContract` will return false for the following
     * types of addresses:
     *
     *  - an externally-owned account
     *  - a contract in construction
     *  - an address where a contract will be created
     *  - an address where a contract lived, but was destroyed
     * ====
     *
     * [IMPORTANT]
     * ====
     * You shouldn't rely on `isContract` to protect against flash loan attacks!
     *
     * Preventing calls from contracts is highly discouraged. It breaks composability, breaks support for smart wallets
     * like Gnosis Safe, and does not provide security since it can be circumvented by calling from a contract
```

```
     * constructor.
     * ====
     */
    function isContract(address account) internal view returns (bool) {
        // This method relies on extcodesize/address.code.length, which returns 0
        // for contracts in construction, since the code is only stored at the end
        // of the constructor execution.

        return account.code.length > 0;
    }

    /**
     * @dev Replacement for Solidity's `transfer`: sends `amount` wei to
     * `recipient`, forwarding all available gas and reverting on errors.
     *
     * https://eips.ethereum.org/EIPS/eip-1884[EIP1884] increases the gas cost
     * of certain opcodes, possibly making contracts go over the 2300 gas limit
     * imposed by `transfer`, making them unable to receive funds via
     * `transfer`. {sendValue} removes this limitation.
     *
     * https://diligence.consensys.net/posts/2019/09/stop-using-soliditys-transfer-now/[Learn more].
     *
     * IMPORTANT: because control is transferred to `recipient`, care must be
     * taken to not create reentrancy vulnerabilities. Consider using
     * {ReentrancyGuard} or the
     *                 https://solidity.readthedocs.io/en/v0.5.11/security-considerations.html#use-the-checks-effects-
interactions-pattern[checks-effects-interactions pattern].
     */
    function sendValue(address payable recipient, uint256 amount) internal {
        require(address(this).balance >= amount, "Address: insufficient balance");

        (bool success, ) = recipient.call{value: amount}("");
        require(success, "Address: unable to send value, recipient may have reverted");
    }

    /**
     * @dev Performs a Solidity function call using a low level `call`. A
     * plain `call` is an unsafe replacement for a function call: use this
```

```
     * function instead.
     *
     * If `target` reverts with a revert reason, it is bubbled up by this
     * function (like regular Solidity function calls).
     *
     * Returns the raw returned data. To convert to the expected return value,
     * use https://solidity.readthedocs.io/en/latest/units-and-global-variables.html?highlight=abi.decode#abi-encoding-and-decoding-functions[`abi.decode`].
     *
     * Requirements:
     *
     * - `target` must be a contract.
     * - calling `target` with `data` must not revert.
     *
     * _Available since v3.1._
     */
    function functionCall(address target, bytes memory data) internal returns (bytes memory) {
        return functionCall(target, data, "Address: low-level call failed");
    }

    /**
     * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`], but with
     * `errorMessage` as a fallback revert reason when `target` reverts.
     *
     * _Available since v3.1._
     */
    function functionCall(
        address target,
        bytes memory data,
        string memory errorMessage
    ) internal returns (bytes memory) {
        return functionCallWithValue(target, data, 0, errorMessage);
    }

    /**
     * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
     * but also transferring `value` wei to `target`.
     *
```

```
     * Requirements:
     *
     * - the calling contract must have an ETH balance of at least `value`.
     * - the called Solidity function must be `payable`.
     *
     * _Available since v3.1._
     */
    function functionCallWithValue(
        address target,
        bytes memory data,
        uint256 value
    ) internal returns (bytes memory) {
        return functionCallWithValue(target, data, value, "Address: low-level call with value failed");
    }

    /**
     * @dev Same as {xref-Address-functionCallWithValue-address-bytes-uint256-}[`functionCallWithValue`], but
     * with `errorMessage` as a fallback revert reason when `target` reverts.
     *
     * _Available since v3.1._
     */
    function functionCallWithValue(
        address target,
        bytes memory data,
        uint256 value,
        string memory errorMessage
    ) internal returns (bytes memory) {
        require(address(this).balance >= value, "Address: insufficient balance for call");
        require(isContract(target), "Address: call to non-contract");

        (bool success, bytes memory returndata) = target.call{value: value}(data);
        return verifyCallResult(success, returndata, errorMessage);
    }

    /**
     * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
     * but performing a static call.
```

```
 *
 * _Available since v3.3._
 */
function functionStaticCall(address target, bytes memory data) internal view returns (bytes memory) {
    return functionStaticCall(target, data, "Address: low-level static call failed");
}

/**
 * @dev Same as {xref-Address-functionCall-address-bytes-string-}[`functionCall`],
 * but performing a static call.
 *
 * _Available since v3.3._
 */
function functionStaticCall(
    address target,
    bytes memory data,
    string memory errorMessage
) internal view returns (bytes memory) {
    require(isContract(target), "Address: static call to non-contract");

    (bool success, bytes memory returndata) = target.staticcall(data);
    return verifyCallResult(success, returndata, errorMessage);
}

/**
 * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
 * but performing a delegate call.
 *
 * _Available since v3.4._
 */
function functionDelegateCall(address target, bytes memory data) internal returns (bytes memory) {
    return functionDelegateCall(target, data, "Address: low-level delegate call failed");
}

/**
 * @dev Same as {xref-Address-functionCall-address-bytes-string-}[`functionCall`],
 * but performing a delegate call.
 *
```

```
 * _Available since v3.4._
 */
function functionDelegateCall(
    address target,
    bytes memory data,
    string memory errorMessage
) internal returns (bytes memory) {
    require(isContract(target), "Address: delegate call to non-contract");

    (bool success, bytes memory returndata) = target.delegatecall(data);
    return verifyCallResult(success, returndata, errorMessage);
}

/**
 * @dev Tool to verifies that a low level call was successful, and revert if it wasn't, either by bubbling the
 * revert reason using the provided one.
 *
 * _Available since v4.3._
 */
function verifyCallResult(
    bool success,
    bytes memory returndata,
    string memory errorMessage
) internal pure returns (bytes memory) {
    if (success) {
        return returndata;
    } else {
        // Look for revert reason and bubble it up if present
        if (returndata.length > 0) {
            // The easiest way to bubble the revert reason is using memory via assembly

            assembly {
                let returndata_size := mload(returndata)
                revert(add(32, returndata), returndata_size)
            }
        } else {
            revert(errorMessage);
        }
```

```
        }
    }
}

// File: @openzeppelin/contracts/token/ERC721/IERC721Receiver.sol


// OpenZeppelin Contracts (last updated v4.6.0) (token/ERC721/IERC721Receiver.sol)


/**
 * @title ERC721 token receiver interface
 * @dev Interface for any contract that wants to support safeTransfers
 * from ERC721 asset contracts.
 */
interface IERC721Receiver {
    /**
     * @dev Whenever an {IERC721} `tokenId` token is transferred to this contract via {IERC721-safeTransferFrom}
     * by `operator` from `from`, this function is called.
     *
     * It must return its Solidity selector to confirm the token transfer.
     * If any other value is returned or the interface is not implemented by the recipient, the transfer will be reverted.
     *
     * The selector can be obtained in Solidity with `IERC721Receiver.onERC721Received.selector`.
     */
    function onERC721Received(
        address operator,
        address from,
        uint256 tokenId,
        bytes calldata data
    ) external returns (bytes4);
}

// File: @openzeppelin/contracts/utils/introspection/IERC165.sol
```

```
// OpenZeppelin Contracts v4.4.1 (utils/introspection/IERC165.sol)


/**
 * @dev Interface of the ERC165 standard, as defined in the
 * https://eips.ethereum.org/EIPS/eip-165[EIP].
 *
 * Implementers can declare support of contract interfaces, which can then be
 * queried by others ({ERC165Checker}).
 *
 * For an implementation, see {ERC165}.
 */
interface IERC165 {
    /**
     * @dev Returns true if this contract implements the interface defined by
     * `interfaceId`. See the corresponding
     * https://eips.ethereum.org/EIPS/eip-165#how-interfaces-are-identified[EIP section]
     * to learn more about how these ids are created.
     *
     * This function call must use less than 30 000 gas.
     */
    function supportsInterface(bytes4 interfaceId) external view returns (bool);
}

// File: @openzeppelin/contracts/utils/introspection/ERC165.sol


// OpenZeppelin Contracts v4.4.1 (utils/introspection/ERC165.sol)


/**
 * @dev Implementation of the {IERC165} interface.
 *
 * Contracts that want to implement ERC165 should inherit from this contract and override {supportsInterface} to check
 * for the additional interface id that will be supported. For example:
 *
 * ```solidity
```

```
 * function supportsInterface(bytes4 interfaceId) public view virtual override returns (bool) {
 *     return interfaceId == type(MyInterface).interfaceId || super.supportsInterface(interfaceId);
 * }
 * ```
 *
 * Alternatively, {ERC165Storage} provides an easier to use but more expensive implementation.
 */
abstract contract ERC165 is IERC165 {
    /**
     * @dev See {IERC165-supportsInterface}.
     */
    function supportsInterface(bytes4 interfaceId) public view virtual override returns (bool) {
        return interfaceId == type(IERC165).interfaceId;
    }
}

// File: @openzeppelin/contracts/token/ERC721/IERC721.sol


// OpenZeppelin Contracts (last updated v4.6.0) (token/ERC721/IERC721.sol)



/**
 * @dev Required interface of an ERC721 compliant contract.
 */
interface IERC721 is IERC165 {
    /**
     * @dev Emitted when `tokenId` token is transferred from `from` to `to`.
     */
    event Transfer(address indexed from, address indexed to, uint256 indexed tokenId);

    /**
     * @dev Emitted when `owner` enables `approved` to manage the `tokenId` token.
     */
    event Approval(address indexed owner, address indexed approved, uint256 indexed tokenId);

    /**
```

```
     * @dev Emitted when `owner` enables or disables (`approved`) `operator` to manage all of its assets.
     */
    event ApprovalForAll(address indexed owner, address indexed operator, bool approved);

    /**
     * @dev Returns the number of tokens in ``owner``'s account.
     */
    function balanceOf(address owner) external view returns (uint256 balance);

    /**
     * @dev Returns the owner of the `tokenId` token.
     *
     * Requirements:
     *
     * - `tokenId` must exist.
     */
    function ownerOf(uint256 tokenId) external view returns (address owner);

    /**
     * @dev Safely transfers `tokenId` token from `from` to `to`.
     *
     * Requirements:
     *
     * - `from` cannot be the zero address.
     * - `to` cannot be the zero address.
     * - `tokenId` token must exist and be owned by `from`.
     * - If the caller is not `from`, it must be approved to move this token by either {approve} or {setApprovalForAll}.
     * - If `to` refers to a smart contract, it must implement {IERC721Receiver-onERC721Received}, which is called upon a safe transfer.
     *
     * Emits a {Transfer} event.
     */
    // function safeTransferFrom(
    //     address from,
    //     address to,
    //     uint256 tokenId,
    //     bytes calldata data
    // ) external;
```

```
    /**
     * @dev Safely transfers `tokenId` token from `from` to `to`, checking first that contract recipients
     * are aware of the ERC721 protocol to prevent tokens from being forever locked.
     *
     * Requirements:
     *
     * - `from` cannot be the zero address.
     * - `to` cannot be the zero address.
     * - `tokenId` token must exist and be owned by `from`.
     * - If the caller is not `from`, it must be have been allowed to move this token by either {approve} or {setApprovalForAll}.
     * - If `to` refers to a smart contract, it must implement {IERC721Receiver-onERC721Received}, which is called upon a safe transfer.
     *
     * Emits a {Transfer} event.
     */
    // function safeTransferFrom(
    //     address from,
    //     address to,
    //     uint256 tokenId
    // ) external;


    /**
     * @dev Transfers `tokenId` token from `from` to `to`.
     *
     * WARNING: Usage of this method is discouraged, use {safeTransferFrom} whenever possible.
     *
     * Requirements:
     *
     * - `from` cannot be the zero address.
     * - `to` cannot be the zero address.
     * - `tokenId` token must be owned by `from`.
     * - If the caller is not `from`, it must be approved to move this token by either {approve} or {setApprovalForAll}.
     *
     * Emits a {Transfer} event.
     */
    function transferFrom(
```

```
    address from,
    address to,
    uint256 tokenId
) external;

/**
 * @dev Gives permission to `to` to transfer `tokenId` token to another account.
 * The approval is cleared when the token is transferred.
 *
 * Only a single account can be approved at a time, so approving the zero address clears previous approvals.
 *
 * Requirements:
 *
 * - The caller must own the token or be an approved operator.
 * - `tokenId` must exist.
 *
 * Emits an {Approval} event.
 */
function approve(address to, uint256 tokenId) external;

/**
 * @dev Approve or remove `operator` as an operator for the caller.
 * Operators can call {transferFrom} or {safeTransferFrom} for any token owned by the caller.
 *
 * Requirements:
 *
 * - The `operator` cannot be the caller.
 *
 * Emits an {ApprovalForAll} event.
 */
function setApprovalForAll(address operator, bool _approved) external;

/**
 * @dev Returns the account approved for `tokenId` token.
 *
 * Requirements:
 *
 * - `tokenId` must exist.
```

```
     */
    function getApproved(uint256 tokenId) external view returns (address operator);

    /**
     * @dev Returns if the `operator` is allowed to manage all of the assets of `owner`.
     *
     * See {setApprovalForAll}
     */
    function isApprovedForAll(address owner, address operator) external view returns (bool);
}

// File: @openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol


// OpenZeppelin Contracts v4.4.1 (token/ERC721/extensions/IERC721Metadata.sol)


/**
 * @title ERC-721 Non-Fungible Token Standard, optional metadata extension
 * @dev See https://eips.ethereum.org/EIPS/eip-721
 */
interface IERC721Metadata is IERC721 {
    /**
     * @dev Returns the token collection name.
     */
    function name() external view returns (string memory);

    /**
     * @dev Returns the token collection symbol.
     */
    function symbol() external view returns (string memory);

    /**
     * @dev Returns the Uniform Resource Identifier (URI) for `tokenId` token.
     */
    function tokenURI(uint256 tokenId) external view returns (string memory);
}
```

```
// File: @openzeppelin/contracts/token/ERC721/ERC721.sol


// OpenZeppelin Contracts (last updated v4.6.0) (token/ERC721/ERC721.sol)









/**
 * @dev Implementation of https://eips.ethereum.org/EIPS/eip-721[ERC721] Non-Fungible Token Standard, including
 * the Metadata extension, but not including the Enumerable extension, which is available separately as
 * {ERC721Enumerable}.
 */
contract ERC721 is Context, ERC165, IERC721, IERC721Metadata {
    using Address for address;
    using Strings for uint256;

    // Token name
    string private _name;

    // Token symbol
    string private _symbol;

    // Mapping from token ID to owner address
    mapping(uint256 => address) private _owners;

    // Mapping owner address to token count
    mapping(address => uint256) private _balances;

    // Mapping from token ID to approved address
    mapping(uint256 => address) private _tokenApprovals;
```

```
    // Mapping from owner to operator approvals
    mapping(address => mapping(address => bool)) private _operatorApprovals;

    /**
     * @dev Initializes the contract by setting a `name` and a `symbol` to the token collection.
     */
    constructor(string memory name_, string memory symbol_) {
        _name = name_;
        _symbol = symbol_;
    }

    /**
     * @dev See {IERC165-supportsInterface}.
     */
    function supportsInterface(bytes4 interfaceId) public view virtual override(ERC165, IERC165) returns (bool)
{
        return
            interfaceId == type(IERC721).interfaceId ||
            interfaceId == type(IERC721Metadata).interfaceId ||
            super.supportsInterface(interfaceId);
    }

    /**
     * @dev See {IERC721-balanceOf}.
     */
    function balanceOf(address owner) public view virtual override returns (uint256) {
        require(owner != address(0), "ERC721: balance query for the zero address");
        return _balances[owner];
    }

    /**
     * @dev See {IERC721-ownerOf}.
     */
    function ownerOf(uint256 tokenId) public view virtual override returns (address) {
        address owner = _owners[tokenId];
        require(owner != address(0), "ERC721: owner query for nonexistent token");
        return owner;
    }
```

```
/**
 * @dev See {IERC721Metadata-name}.
 */
function name() public view virtual override returns (string memory) {
    return _name;
}

/**
 * @dev See {IERC721Metadata-symbol}.
 */
function symbol() public view virtual override returns (string memory) {
    return _symbol;
}

/**
 * @dev See {IERC721Metadata-tokenURI}.
 */
function tokenURI(uint256 tokenId) public view virtual override returns (string memory) {
    require(_exists(tokenId), "ERC721Metadata: URI query for nonexistent token");

    string memory baseURI = _baseURI();
    return bytes(baseURI).length > 0 ? string(abi.encodePacked(baseURI, tokenId.toString())) : "";
}

/**
 * @dev Base URI for computing {tokenURI}. If set, the resulting URI for each
 * token will be the concatenation of the `baseURI` and the `tokenId`. Empty
 * by default, can be overridden in child contracts.
 */
function _baseURI() internal view virtual returns (string memory) {
    return "";
}

/**
 * @dev See {IERC721-approve}.
 */
function approve(address to, uint256 tokenId) public virtual override {
```

```
    address owner = ERC721.ownerOf(tokenId);
    require(to != owner, "ERC721: approval to current owner");

    require(
        _msgSender() == owner || isApprovedForAll(owner, _msgSender()),
        "ERC721: approve caller is not owner nor approved for all"
    );

    _approve(to, tokenId);
}

/**
 * @dev See {IERC721-getApproved}.
 */
function getApproved(uint256 tokenId) public view virtual override returns (address) {
    require(_exists(tokenId), "ERC721: approved query for nonexistent token");

    return _tokenApprovals[tokenId];
}

/**
 * @dev See {IERC721-setApprovalForAll}.
 */
function setApprovalForAll(address operator, bool approved) public virtual override {
    _setApprovalForAll(_msgSender(), operator, approved);
}

/**
 * @dev See {IERC721-isApprovedForAll}.
 */
function isApprovedForAll(address owner, address operator) public view virtual override returns (bool) {
    return _operatorApprovals[owner][operator];
}

/**
 * @dev See {IERC721-transferFrom}.
 */
function transferFrom(
```

```
        address from,
        address to,
        uint256 tokenId
    ) public virtual override {
        //solhint-disable-next-line max-line-length
        require(_isApprovedOrOwner(_msgSender(), tokenId), "ERC721: transfer caller is not owner nor approved");

        _transfer(from, to, tokenId);
    }

    /**
     * @dev See {IERC721-safeTransferFrom}.
     */
    // function safeTransferFrom(
    //     address from,
    //     address to,
    //     uint256 tokenId
    // ) public virtual override {
    //     safeTransferFrom(from, to, tokenId, "");
    // }

    /**
     * @dev See {IERC721-safeTransferFrom}.
     */
    // function safeTransferFrom(
    //     address from,
    //     address to,
    //     uint256 tokenId,
    //     bytes memory _data
    // ) public virtual override {
    //         require(_isApprovedOrOwner(_msgSender(), tokenId), "ERC721: transfer caller is not owner nor approved");
    //     _safeTransfer(from, to, tokenId, _data);
    // }

    /**
     * @dev Safely transfers `tokenId` token from `from` to `to`, checking first that contract recipients
     * are aware of the ERC721 protocol to prevent tokens from being forever locked.
```

```
    *
    * `_data` is additional data, it has no specified format and it is sent in call to `to`.
    *
    * This internal function is equivalent to {safeTransferFrom}, and can be used to e.g.
    * implement alternative mechanisms to perform token transfer, such as signature-based.
    *
    * Requirements:
    *
    * - `from` cannot be the zero address.
    * - `to` cannot be the zero address.
    * - `tokenId` token must exist and be owned by `from`.
    * - If `to` refers to a smart contract, it must implement {IERC721Receiver-onERC721Received}, which is called upon a safe transfer.
    *
    * Emits a {Transfer} event.
    */
   function _safeTransfer(
      address from,
      address to,
      uint256 tokenId,
      bytes memory _data
   ) internal virtual {
      _transfer(from, to, tokenId);
      require(_checkOnERC721Received(from, to, tokenId, _data), "ERC721: transfer to non ERC721Receiver implementer");
   }


   /**
    * @dev Returns whether `tokenId` exists.
    *
    * Tokens can be managed by their owner or approved accounts via {approve} or {setApprovalForAll}.
    *
    * Tokens start existing when they are minted (`_mint`),
    * and stop existing when they are burned (`_burn`).
    */
   function _exists(uint256 tokenId) internal view virtual returns (bool) {
      return _owners[tokenId] != address(0);
   }
```

```
    /**
     * @dev Returns whether `spender` is allowed to manage `tokenId`.
     *
     * Requirements:
     *
     * - `tokenId` must exist.
     */
    function _isApprovedOrOwner(address spender, uint256 tokenId) internal view virtual returns (bool) {
        require(_exists(tokenId), "ERC721: operator query for nonexistent token");
        address owner = ERC721.ownerOf(tokenId);
        return (spender == owner || isApprovedForAll(owner, spender) || getApproved(tokenId) == spender);
    }

    /**
     * @dev Safely mints `tokenId` and transfers it to `to`.
     *
     * Requirements:
     *
     * - `tokenId` must not exist.
     * - If `to` refers to a smart contract, it must implement {IERC721Receiver-onERC721Received}, which is called upon a safe transfer.
     *
     * Emits a {Transfer} event.
     */
    function _safeMint(address to, uint256 tokenId) internal virtual {
        _safeMint(to, tokenId, "");
    }

    /**
     * @dev Same as {xref-ERC721-_safeMint-address-uint256-}[`_safeMint`], with an additional `data` parameter which is
     * forwarded in {IERC721Receiver-onERC721Received} to contract recipients.
     */
    function _safeMint(
        address to,
        uint256 tokenId,
        bytes memory _data
    ) internal virtual {
```

```
    _mint(to, tokenId);
    require(
        _checkOnERC721Received(address(0), to, tokenId, _data),
        "ERC721: transfer to non ERC721Receiver implementer"
    );
}

/**
 * @dev Mints `tokenId` and transfers it to `to`.
 *
 * WARNING: Usage of this method is discouraged, use {_safeMint} whenever possible
 *
 * Requirements:
 *
 * - `tokenId` must not exist.
 * - `to` cannot be the zero address.
 *
 * Emits a {Transfer} event.
 */
function _mint(address to, uint256 tokenId) internal virtual {
    require(to != address(0), "ERC721: mint to the zero address");
    require(!_exists(tokenId), "ERC721: token already minted");

    _beforeTokenTransfer(address(0), to, tokenId);

    _balances[to] += 1;
    _owners[tokenId] = to;

    emit Transfer(address(0), to, tokenId);

    _afterTokenTransfer(address(0), to, tokenId);
}

/**
 * @dev Destroys `tokenId`.
 * The approval is cleared when the token is burned.
 *
 * Requirements:
```

```
 *
 * - `tokenId` must exist.
 *
 * Emits a {Transfer} event.
 */
function _burn(uint256 tokenId) internal virtual {
    address owner = ERC721.ownerOf(tokenId);

    _beforeTokenTransfer(owner, address(0), tokenId);

    // Clear approvals
    _approve(address(0), tokenId);

    _balances[owner] -= 1;
    delete _owners[tokenId];

    emit Transfer(owner, address(0), tokenId);

    _afterTokenTransfer(owner, address(0), tokenId);
}

/**
 * @dev Transfers `tokenId` from `from` to `to`.
 *  As opposed to {transferFrom}, this imposes no restrictions on msg.sender.
 *
 * Requirements:
 *
 * - `to` cannot be the zero address.
 * - `tokenId` token must be owned by `from`.
 *
 * Emits a {Transfer} event.
 */
function _transfer(
    address from,
    address to,
    uint256 tokenId
) internal virtual {
    require(ERC721.ownerOf(tokenId) == from, "ERC721: transfer from incorrect owner");
```

```
    require(to != address(0), "ERC721: transfer to the zero address");

    _beforeTokenTransfer(from, to, tokenId);

    // Clear approvals from the previous owner
    _approve(address(0), tokenId);

    _balances[from] -= 1;
    _balances[to] += 1;
    _owners[tokenId] = to;

    emit Transfer(from, to, tokenId);

    _afterTokenTransfer(from, to, tokenId);
}

/**
 * @dev Approve `to` to operate on `tokenId`
 *
 * Emits a {Approval} event.
 */
function _approve(address to, uint256 tokenId) internal virtual {
    _tokenApprovals[tokenId] = to;
    emit Approval(ERC721.ownerOf(tokenId), to, tokenId);
}

/**
 * @dev Approve `operator` to operate on all of `owner` tokens
 *
 * Emits a {ApprovalForAll} event.
 */
function _setApprovalForAll(
    address owner,
    address operator,
    bool approved
) internal virtual {
    require(owner != operator, "ERC721: approve to caller");
    _operatorApprovals[owner][operator] = approved;
```

```
        emit ApprovalForAll(owner, operator, approved);
    }

    /**
     * @dev Internal function to invoke {IERC721Receiver-onERC721Received} on a target address.
     * The call is not executed if the target address is not a contract.
     *
     * @param from address representing the previous owner of the given token ID
     * @param to target address that will receive the tokens
     * @param tokenId uint256 ID of the token to be transferred
     * @param _data bytes optional data to send along with the call
     * @return bool whether the call correctly returned the expected magic value
     */
    function _checkOnERC721Received(
        address from,
        address to,
        uint256 tokenId,
        bytes memory _data
    ) private returns (bool) {
        if (to.isContract()) {
            try IERC721Receiver(to).onERC721Received(_msgSender(), from, tokenId, _data) returns (bytes4 retval) {
                return retval == IERC721Receiver.onERC721Received.selector;
            } catch (bytes memory reason) {
                if (reason.length == 0) {
                    revert("ERC721: transfer to non ERC721Receiver implementer");
                } else {
                    assembly {
                        revert(add(32, reason), mload(reason))
                    }
                }
            }
        } else {
            return true;
        }
    }

    /**
```

```
     * @dev Hook that is called before any token transfer. This includes minting
     * and burning.
     *
     * Calling conditions:
     *
     * - When `from` and `to` are both non-zero, ``from``'s `tokenId` will be
     * transferred to `to`.
     * - When `from` is zero, `tokenId` will be minted for `to`.
     * - When `to` is zero, ``from``'s `tokenId` will be burned.
     * - `from` and `to` are never both zero.
     *
     * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
     */
    function _beforeTokenTransfer(
        address from,
        address to,
        uint256 tokenId
    ) internal virtual {}

    /**
     * @dev Hook that is called after any transfer of tokens. This includes
     * minting and burning.
     *
     * Calling conditions:
     *
     * - when `from` and `to` are both non-zero.
     * - `from` and `to` are never both zero.
     *
     * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
     */
    function _afterTokenTransfer(
        address from,
        address to,
        uint256 tokenId
    ) internal virtual {}
}

// File: @openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol
```

```
// OpenZeppelin Contracts v4.4.1 (token/ERC721/extensions/ERC721URIStorage.sol)



/**
 * @dev ERC721 token with storage based token URI management.
 */
abstract contract ERC721URIStorage is ERC721 {
    using Strings for uint256;

    // Optional mapping for token URIs
    mapping(uint256 => string) private _tokenURIs;

    /**
     * @dev See {IERC721Metadata-tokenURI}.
     */
    function tokenURI(uint256 tokenId) public view virtual override returns (string memory) {
        require(_exists(tokenId), "ERC721URIStorage: URI query for nonexistent token");

        string memory _tokenURI = _tokenURIs[tokenId];
        string memory base = _baseURI();

        // If there is no base URI, return the token URI.
        if (bytes(base).length == 0) {
            return _tokenURI;
        }
        // If both are set, concatenate the baseURI and tokenURI (via abi.encodePacked).
        if (bytes(_tokenURI).length > 0) {
            return string(abi.encodePacked(base, _tokenURI));
        }

        return super.tokenURI(tokenId);
    }

    /**
     * @dev Sets `_tokenURI` as the tokenURI of `tokenId`.
```

```
     *
     * Requirements:
     *
     * - `tokenId` must exist.
     */
    function _setTokenURI(uint256 tokenId, string memory _tokenURI) internal virtual {
        require(_exists(tokenId), "ERC721URIStorage: URI set of nonexistent token");
        _tokenURIs[tokenId] = _tokenURI;
    }

    /**
     * @dev Destroys `tokenId`.
     * The approval is cleared when the token is burned.
     *
     * Requirements:
     *
     * - `tokenId` must exist.
     *
     * Emits a {Transfer} event.
     */
    function _burn(uint256 tokenId) internal virtual override {
        super._burn(tokenId);

        if (bytes(_tokenURIs[tokenId]).length != 0) {
            delete _tokenURIs[tokenId];
        }
    }
}
// File: contracts/SoulToken.sol

contract UniversityDegree is ERC721URIStorage  {

    address owner;

    using Counters for Counters.Counter;
    Counters.Counter private _tokenIds;

    constructor() ERC721("SomeNFT", "CoolNFT") {
```

```
        owner = msg.sender;
    }

    mapping(address => string) public personToDegree;

    function claimDegree(string memory tokenURI)
        public
        returns (uint256)
    {
        require(issuedDegrees[msg.sender], "Degree is not issued");

        _tokenIds.increment();

        uint256 newItemId = _tokenIds.current();
        _mint(msg.sender, newItemId);
        _setTokenURI(newItemId, tokenURI);

        personToDegree[msg.sender] = tokenURI;
        issuedDegrees[msg.sender] = false;

        return newItemId;
    }

    function checkDegreeOfPerson(address person) external view returns (string memory) {
        return personToDegree[person];
    }

    modifier onlyOwner() {
        require(msg.sender == owner);
        _;
    }

    mapping(address => bool) public issuedDegrees;

    function issueDegree(address to) onlyOwner external {
        issuedDegrees[to] = true;
    }
}
```

{
"image":"https://ipfs.io/ipfs/QmPdE1ngwZuDSvxf7pbqj6VruDD5c7Nw9GpDayZgNeyoEs",
"description": "´Nota obtenida en la asignatura X del Grado Y de la Universidad de Sevilla",
"attributes":
[
{
"trait_type": "Nota",
"value": 7,
"max-value": 10
},
{
"trait_type": "Grado",
"value": "Grado en Ingenieria de Y"
},
{
"trait_type": "Nombre Alumno",
"value": "Fulanito Perez"
}
]
}